Matplotlib custom tools universe

https://github.com/fariza/pycon2017

Optical specialist @ Matrox

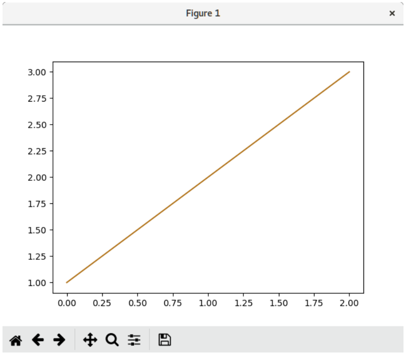
November 19, 2017

Who is this for?

- Do you play with a lot of data?
- Do you plot?
- Do you plot a lot of data?
- Do you want to get your hands dirty?
- You didn't have anything better to do?

The gui

- Key-only tools
  - Grid
  - log
  - ...
- Toolbar buttons
  - Home
  - save
  - ...



Today

Key press events

- Single function (Huge, ugly)

```
def key_press_handler(event, canvas, toolbar=None)
```

Today

Key press events

- Single function (Huge, ugly)

  ```
  def key_press_handler(event, canvas, toolbar=None)
  ```

- Some events are transmited to the toolbar

Today

Key press events

- Single function (Huge, ugly)

  ```
  def key_press_handler(event, canvas, toolbar=None)
  ```

- Some events are transmited to the toolbar
- If no toolbar, then some events are not available

Today

Key press events

- Single function (Huge, ugly)

  ```
  def key_press_handler(event, canvas, toolbar=None)
  ```
- Some events are transmited to the toolbar
- If no toolbar, then some events are not available
- Some events are handled in place, without possibility for for the Toolbar to add a "button"

Today

Key press events

- Single function (Huge, ugly)

  ```
  def key_press_handler(event, canvas, toolbar=None)
  ```

- Some events are transmited to the toolbar
- If no toolbar, then some events are not available
- Some events are handled in place, without possibility for for the Toolbar to add a "button"
- No easy way to add new key-event handlers

Today

Key press events

- Single function (Huge, ugly)

  ```
  def key_press_handler(event, canvas, toolbar=None)
  ```
- Some events are transmited to the toolbar
- If no toolbar, then some events are not available
- Some events are handled in place, without possibility for for the Toolbar to add a "button"
- No easy way to add new key-event handlers
- No way to know the associated keys

Today

Toolbar

- Base class defines most of the handling

Today

Toolbar

- Base class defines most of the handling
- Backend specific code

Today

Toolbar

- Base class defines most of the handling
- Backend specific code
- Gui and process is done in the same place

Today

Toolbar

- Base class defines most of the handling
- Backend specific code
- Gui and process is done in the same place
- No "easy" way to call methods inside the toolbar

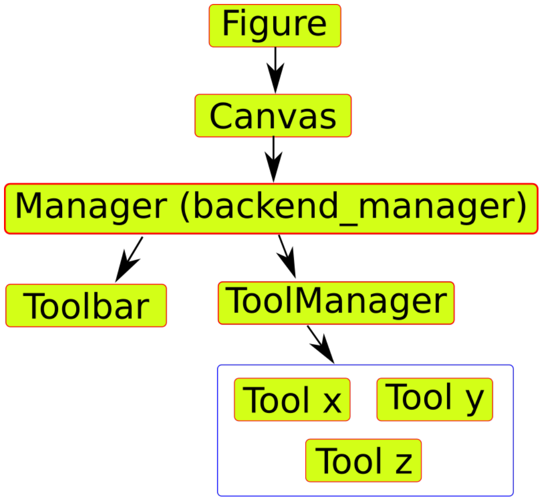
Today

Toolbar

- Base class defines most of the handling
- Backend specific code
- Gui and process is done in the same place
- No "easy" way to call methods inside the toolbar
- No way to add new tools without writing a new backend

New age

Where to find things



New age

ToolManager

- Add/remove/modify tools dynamically

New age

ToolManager

- Add/remove/modify tools dynamically
- Does not have any gui related code

New age

ToolManager

- Add/remove/modify tools dynamically
- Does not have any gui related code
- Keep track of associated keys

New age

ToolManager

- Add/remove/modify tools dynamically
- Does not have any gui related code
- Keep track of associated keys
- Manage tool radio groups

New age

Tools

- Two kind of tools:

New age

Tools

- Two kind of tools:
    - Basic tool

New age

Tools

- Two kind of tools:
  - Basic tool
  - Toggle tool

New age

Toolbar

- Has no logic related to tools

New age

Toolbar

- Has no logic related to tools
- Two important methods: addtool, removetool

New age

Toolbar

- Has no logic related to tools
- Two important methods: addtool, removetool
- Hooks to tool-events to change button state if a tool is triggered by any way other than "click"

New age

Toolbar

- Has no logic related to tools
- Two important methods: addtool, removetool
- Hooks to tool-events to change button state if a tool is triggered by any way other than "click"
- Simple backend creation

Does it work?

Let's check

```
import matplotlib
# matplotlib.use('GTK3Agg')
matplotlib.use('tkAgg')
matplotlib.rcParams['toolbar'] = 'toolmanager'
import matplotlib.pyplot as plt

fig = plt.figure()
plt.plot([1, 2, 3], label='Super data')
plt.show()
```

Play with buttons

Remove one button

```
fig.canvas.manager.toolbar.remove_toolitem('forward')
```

Play with buttons

Remove one button

```
fig.canvas.manager.toolbar.remove_toolitem('forward')
```

Duplicate a button

```
fig.canvas.manager.toolbar.add_tool('zoom', 'foo')
```

Play with buttons

Remove one button

```
fig.canvas.manager.toolbar.remove_toolitem('forward')
```

Duplicate a button

```
fig.canvas.manager.toolbar.add_tool('zoom', 'foo')
```

Completely remove one button

```
fig.canvas.manager.toolmanager.remove_tool('save')
```

Simple tool

Extra-Simple tool

```
from matplotlib.backend_tools import ToolBase


class ExtraSimple(ToolBase):
    description = 'Encourage yourself'
    default_keymap = 'C'

    def trigger(self, *args, **kwargs):
        self.toolmanager.message_event("You are doing great!!")
```

Simple tool

Extra-Simple tool

```
from matplotlib.backend_tools import ToolBase


class ExtraSimple(ToolBase):
    description = 'Encourage yourself'
    default_keymap = 'C'

    def trigger(self, *args, **kwargs):
        self.toolmanager.message_event("You are doing great!!")
```

Add the tool to toolmanager

```
fig.canvas.manager.toolmanager.add_tool('simple', ExtraSimple)
```

Simple tool

Extra-Simple tool

```
from matplotlib.backend_tools import ToolBase


class ExtraSimple(ToolBase):
    description = 'Encourage yourself'
    default_keymap = 'C'

    def trigger(self, *args, **kwargs):
        self.toolmanager.message_event("You are doing great!!")
```

Add the tool to toolmanager

```
fig.canvas.manager.toolmanager.add_tool('simple', ExtraSimple)
```

Add to the toolbar

```
fig.canvas.manager.toolbar.add_tool('simple', 'navigation')
```

Toggle tool

Toggle legend

```python
from matplotlib.backend_tools import ToolToggleBase

class ToggleLegend(ToolToggleBase):
    description = 'Toggle the legend'
    default_toggled = True
    default_keymap = 'l'

    def visibility(self, state):
        for leg in list(self.figure.legends):
            leg.set_visible(state)
        for a in self.figure.get_axes():
            leg = a.get_legend()
            if leg:
                leg.set_visible(state)
        self.figure.canvas.draw_idle()

    def enable(self, event):
        self.visibility(True)

    def disable(self, event):
        self.visibility(False)
```

Toggle tool

Add a legend

```
plt.legend()
```

Toggle tool

Add a legend

```
plt.legend()
```

Add the tool

```
fig.canvas.manager.toolmanager.add_tool('legend', ToggleLegend)
fig.canvas.manager.toolbar.add_tool('legend', 'io')
```