Grafiska användargränssnitt

Daniel Bosk

KTH EECS

7th October 2020

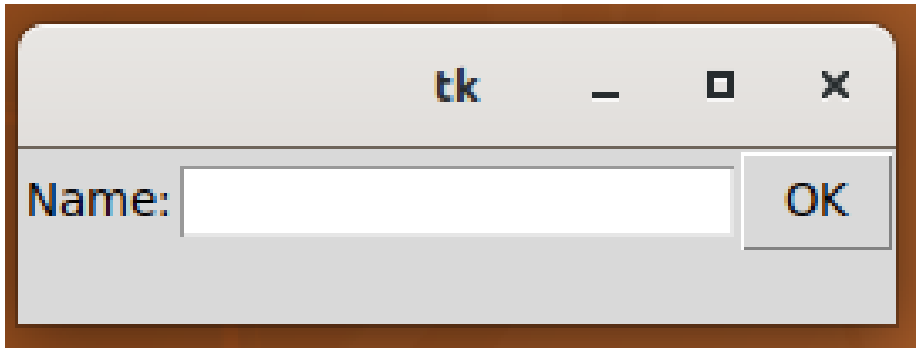
Händelsebaserat

- Oändlig slinga som läser meddelanden från OS.
- OS skickar ett meddelande per händelse.

helloworld.py

```
1  """Hello world with tkinter"""
2
3  import tkinter as tk
4
5  def main():
6      """Test program"""
7      root = tk.Tk()
8      text = tk.Label(root, text="Hello, world!")
9      text.pack()
10     root.mainloop()
11
12 if __name__ == "__main__":
13     main()
```

Figure: Fönstret som genereras av helloworld.py.



(a) Ett program med textfält för inmatning.

(b) Samma program med inmatning och utmatning.

Figure: Skärmdumpar av ett grafiskt gränssnitt med in- och utmatning.

hello_user_bad.py

```
7  window = tk.Tk()
8
9  # Name:
10 name_label = tk.Label(window, text="Name:")
11 name_label.grid(row=0, column=0)
12
13 # Input field
14 name = tk.Entry(window)
15 name.grid(row=0, column=1)
16
17 # Output below
18 output = tk.Label(window)
19 output.grid(row=1, column=0)
```

hello_user_bad.py

```
21  # Don't do this!
22  def update_output():
23      output["text"] = f"Hello, {name.get()}!"
24
25  # OK button next to input field
26  ok_button = tk.Button(window, text="OK",
27                        command=update_output)
28  ok_button.grid(row=0, column=2)
29
30  # Run the window --- loop forever
31  window.mainloop()
```

Note

- Vi kan använda klasser istället!

hello_user_good.py

```
1  """Super pretty program"""
2
3  import tkinter as tk
4
5  class InputGUI:
6      """GUI which takes name input and prints 'Hello, {name}!'"""
7      def __init__(self):
8          self.window = tk.Tk()
9
10         # Name:
11         self.name_label = tk.Label(self.window, text="Name:")
12         self.name_label.grid(row=0, column=0)
```

hello_user_good.py

```
18          # OK button next to input field
19          self.ok_button = tk.Button(self.window, text="OK",
20                                     command=self.update_output)
21          self.ok_button.grid(row=0, column=2)
22
23          # Output below
24          self.output = tk.Label(self.window)
25          self.output.grid(row=1, column=0)

...

30      def update_output(self):
31          """Update the output label to what's in the name entry"""
32          self.output["text"] = f"Hello, {self.name.get()}!"
```

hello_user_counter.py

```
5   class InputGUI:
6       """GUI which takes name input and prints 'Hello, {name}!'"""
7       def __init__(self):
8           self.window = tk.Tk()
9
10          self.counter = 0
11
12          # Name:
13          self.name_label = tk.Label(self.window, text="Name:")
14          self.name_label.grid(row=0, column=0)

...

32      def update_output(self):
33          """Update the output label to what's in the name entry"""
34          self.output["text"] = f"Hello, {self.name.get()}" + "!"*self.co
35          self.counter += 1
```

hello_user_better.py

```
5   class InputGUI(tk.Tk):
6       """GUI which takes name input and prints 'Hello, {name}!'"""
7       def __init__(self):
8           # Initialize tk.Tk itself
9           super().__init__()
10
11          # Name:
12          self.name_label = tk.Label(text="Name:")
13          self.name_label.grid(row=0, column=0)

    ...

33  def main():
34      """Test program"""
35      window = InputGUI()
36
37      # Run the window --- loop forever
38      window.mainloop()
```

hello_user_even_better.py

```
5   class InputGUI(tk.Tk):
6       """GUI which takes name input and prints 'Hello, {name}!'"""
7       def __init__(self):
8           # Initialize tk.Tk itself
9           super().__init__()

...

24          self.output = tk.StringVar()
25
26          # Output below
27          self.output_label = tk.Label(textvariable=self.output)
28          self.output_label.grid(row=1)
29
30      def update_output(self):
31          """Update the output label to what's in the name entry"""
32          self.output.set(f"Hello, {self.name.get()}!")
```

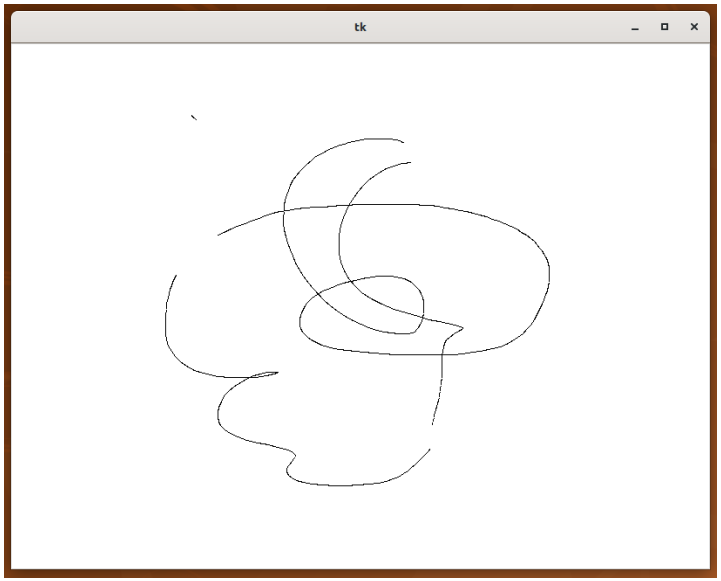


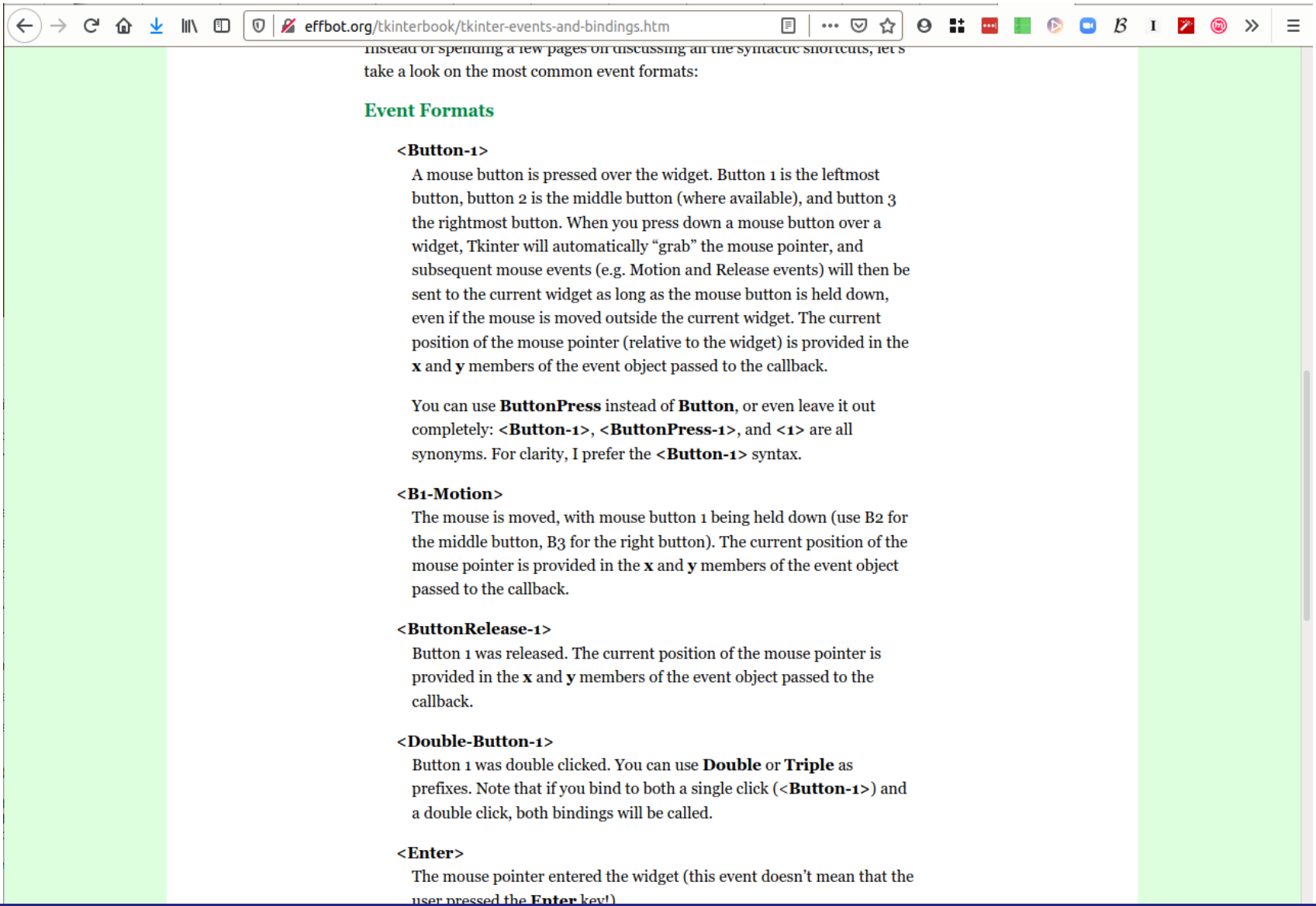
Figure: Skärmdump av program där man ritar med musen.

draw.py

```
5   class DrawGUI(tk.Tk):
6       """GUI which takes name input and prints 'Hello, {name}!'"""
7       def __init__(self):
8           # Initialize tk.Tk itself
9           super().__init__()
10
11          # Create a large canvas to draw on
12          self.canvas = tk.Canvas(width=800, height=600, bg="white")
13          self.canvas.pack()

    ...

18          # We need to check when
19          # ... mouse moves with button pressed
20          # ... when button released
21          self.canvas.bind("<B1-Motion>", self.paint)
22          self.canvas.bind("<ButtonRelease-1>", self.reset_old_coord)
```


effbot.org/tkinterbook/tkinter-events-and-bindings.htm

Instead of spending a few pages on discussing all the syntactic shortcuts, let's take a look on the most common event formats:

Event Formats

<Button-1>
A mouse button is pressed over the widget. Button 1 is the leftmost button, button 2 is the middle button (where available), and button 3 the rightmost button. When you press down a mouse button over a widget, Tkinter will automatically "grab" the mouse pointer, and subsequent mouse events (e.g. Motion and Release events) will then be sent to the current widget as long as the mouse button is held down, even if the mouse is moved outside the current widget. The current position of the mouse pointer (relative to the widget) is provided in the **x** and **y** members of the event object passed to the callback.

You can use **ButtonPress** instead of **Button**, or even leave it out completely: **<Button-1>**, **<ButtonPress-1>**, and **<1>** are all synonyms. For clarity, I prefer the **<Button-1>** syntax.

<B1-Motion>
The mouse is moved, with mouse button 1 being held down (use B2 for the middle button, B3 for the right button). The current position of the mouse pointer is provided in the **x** and **y** members of the event object passed to the callback.

<ButtonRelease-1>
Button 1 was released. The current position of the mouse pointer is provided in the **x** and **y** members of the event object passed to the callback.

<Double-Button-1>
Button 1 was double clicked. You can use **Double** or **Triple** as prefixes. Note that if you bind to both a single click (**<Button-1>**) and a double click, both bindings will be called.

<Enter>
The mouse pointer entered the widget (this event doesn't mean that the user pressed the **Enter** key!)

draw.py

```
24  def paint(self, event):
25      """ Draw a line between two points """
26      # If we don't have old coordinates, we must wait for new
27      if self.x_old is None:
28          self.x_old = event.x
29          self.y_old = event.y
30          return
31
32      # If we have old coordinates, we can draw a line
33      x_start, y_start = self.x_old, self.y_old
34      x_dest, y_dest = self.x_old, self.y_old = event.x, event.y
35      self.canvas.create_line(x_start, y_start,
36                              x_dest, y_dest)
37
38  def reset_old_coord(self, _):
39      """Reset x_old and y_old to None"""
40      self.x_old = self.y_old = None
```

Annan typ av inmatning

effbot.org/tkinterbook/canvas.htm#Tkinter.Canvas.create_line-method

create_line(coords, **options) [#]

Draws a line on the canvas.

coords
Image coordinates.
***options*
Line options.
activedash=
activefill=
Line color to use when the mouse pointer is moved over the item, if different from **fill**.
activestipple=
activewidth=
Default is 0.0.
arrow=
Default is NONE.
arrowshape=
Default is “8 10 3”.
capstyle=
Default is BUTT.
dash=
Dash pattern, given as a list of segment lengths. Only the odd segments are drawn.
dashoffset=
Default is 0.
disableddash=
disabledfill=
Line color to use when the item is disabled, if different from **fill**.
disabledstipple=
disabledwidth=
Default is 0.0.
fill=
Line color. Default is “black”.
joinstyle=
Default is ROUND.

Question

- Hur mycket meddelanden/händelser hanterar vi?

Solution

- *Vi kan testa med draw_debug.py.*

Question

- Hur mycket meddelanden/händelser hanterar vi?

Solution

- *Vi kan testa med draw_debug.py.*

Exercise

- Hur kan vi lägga till knappar för att byta färg?

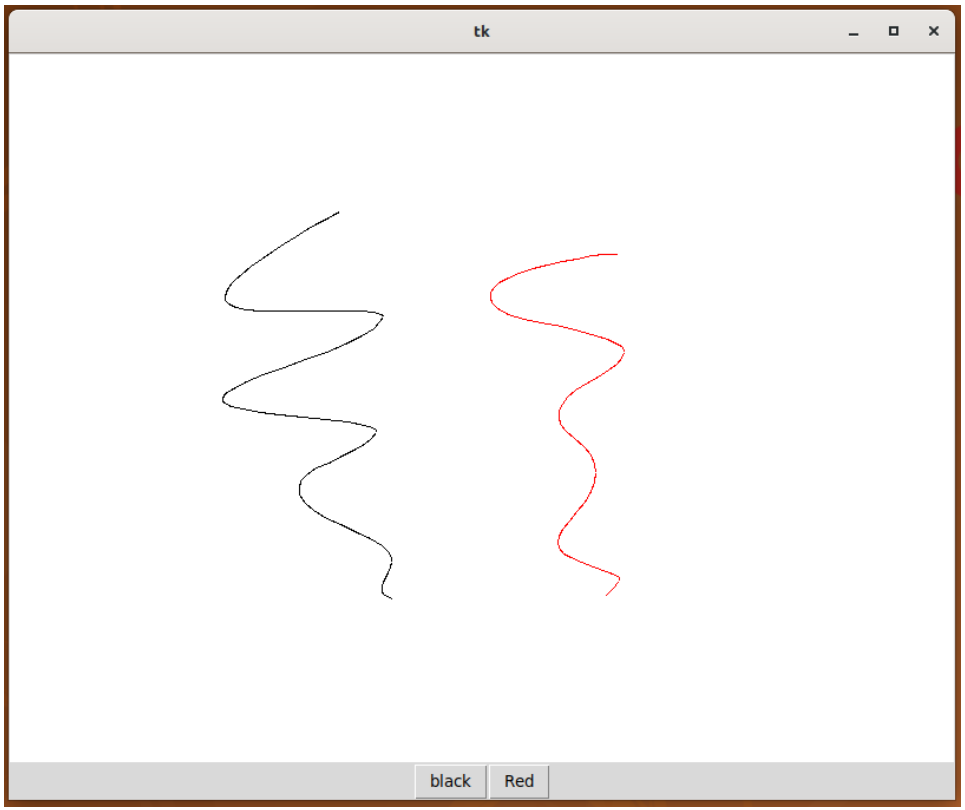


Figure: Ett ritgränssnitt med knappar för att byta färg.

draw_colors.py

```
5   class DrawGUI(tk.Tk):
6       """GUI which takes name input and prints 'Hello, {name}!'"""
7       def __init__(self):

...

27          # We want to group the color buttons in a frame
28          self.color_frame = tk.Frame()
29          # We want the frame at the bottom, so we can pack it
30          self.color_frame.pack()
31
32          self.black_button = tk.Button(self.color_frame, text="black",
33                                        command=self.set_black)
34          self.black_button.grid(row=0, column=0)
```

draw_colors.py

```
40      def paint(self, event):
...
50          # If we have old coordinates, we can draw a line
51          x_start, y_start = self.x_old, self.y_old
52          x_dest, y_dest = self.x_old, self.y_old = event.x, event.y
53          self.canvas.create_line(x_start, y_start,
54                                  x_dest, y_dest, fill=self.color)
...
61      def set_black(self):
62          """Change line color to black"""
63          print("change to black")
64          self.color = "black"
```

Annan typ av inmatning