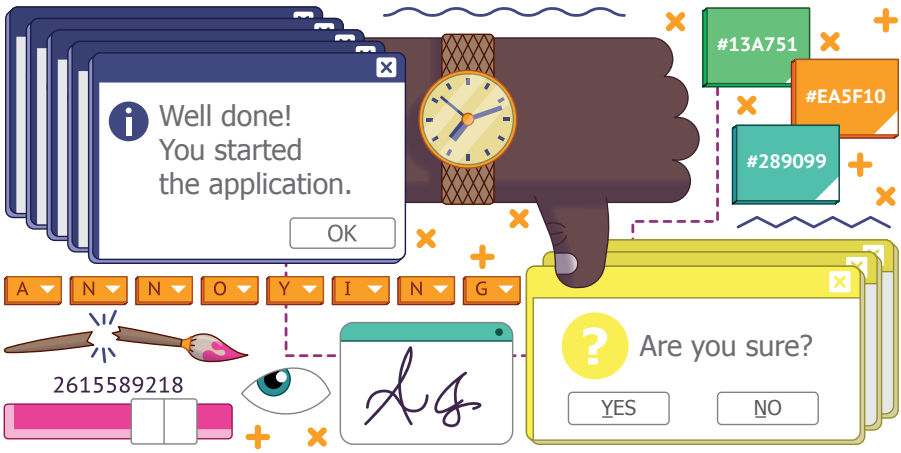
Chapter 5

World's Worst GUI

Learn good GUI design by doing it all wrong first!



ts time to really go to town with your GUIs and experiment with different widgets, colours, fonts, and features. Like most experiments, it's likely that you won't get it right first time! In fact, you are going to explore the wrong way to approach creating your GUI.

It's hard to read

The right choice of GUI colour and font are important. It's important that the contrast between background and text colour ensure that your GUI is easily readable. What you shouldn't do it is use two very similar colours.

Import the widgets at the top of the code:

```
from guizero import App, Text
```

Create an app with a title:

```
app = App("it's all gone wrong")
title = Text(app, text="Some hard to read text")
```

```
app.display()
```

Experiment by changing the colours, font, and text size (see **worst1.py** listing, page 41). My choices are not the best!

```
app = App("it's all gone wrong", bg="dark green")
title = Text(app, text="Some hard-to-read text", size="14",
font="Comic Sans", color="green")
```

It's important that text on a GUI also stays around long enough to be read. It certainly shouldn't disappear or start flashing.

All widgets in guizero can be made invisible (or visible again) using the `hide()` and `show()` functions. Using the `repeat` function in guizero to run a function every second, you can make your text hide and show itself and appear to flash.

Create a function which will hide the text if it's visible and show it if it's not:

```
def flash_text():
    if title.visible:
        title.hide()
    else:
        title.show()
```

Before the app is displayed, use `repeat` to make the `flash_text` function run every 1000 milliseconds (1 second).

```
app.repeat(1000, flash_text)

app.display()
```

Your code should now look like **worst2.py**. Test your app: the title text should flash, appearing and disappearing once every second.

The wrong widget

Using an appropriate widget can be the difference between a great GUI and one which is completely unusable.

Which widget would you use to enter a date? A TextBox? Multiple Combos? A TextBox would be more flexible but would require validation and formatting. Multiple Combos for year, month, and day wouldn't require validating but would be slower to use.

Figure 1 A slider to set date and time

Figure 2 Combos to choose letters

Using a Slider to set a date and time (**Figure 1**), as in the **worst3.py** code example, is not a great idea, though.

The Slider widget returns a number between 0 and 999,999,999. This is the number of seconds since 1 January 1970. The function `ctime()` is used to turn this number into a date and time.

Getting text from your user is simple: a TextBox or a multi-line TextBox should fulfil all your needs. Is it too simple, though. Does this require too much typing?

What about the user who just wants to use a mouse? Perhaps a series of Combos each containing all the letters in the alphabet would be better (**Figure 2**)?

Start by importing the guizero widgets and `ascii_letters`.

```
from guizero import App, Combo
from string import ascii_letters
```

`ascii_letters` is a list containing all the 'printable' ASCII characters which you can use as the options for the Combo.

Create a single Combo which contains all the letters and displays the app.

```
a_letter = Combo(app, options=" " + ascii_letters, align="left")

app.display()
```

Your program should now resemble **worst4.py**. Running it, you will see a single Combo which contains all the letters plus a space and is aligned to the left of the window.

To get a line of letters together, you could continually add Combo widgets to your app, e.g.:

```
a_letter = Combo(app, options=" " + ascii_letters, align="left")
b_letter = Combo(app, options=" " + ascii_letters, align="left")
c_letter = Combo(app, options=" " + ascii_letters, align="left")
```

By aligning each Combo widget to the left, the widgets are displayed next to each other against the left edge.

Alternatively, you could use a `for` loop, create a list of letters, and append each letter to the list, as shown in **worst5.py**.

Try both these approaches and see which you prefer. The `for` loop is more flexible as it allows you to create as many letters as you like.

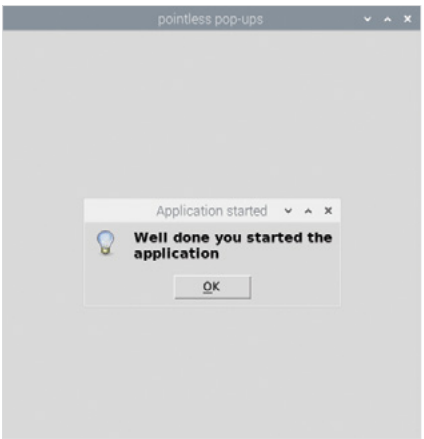


Figure 3 Pointless pop-up

Pop-ups

No terrible GUI would be complete without a pop-up box. guizero contains a number of pop-up boxes, which can be used to let users know something important or gather useful information. They can also be used to irritate and annoy users!

First, create an application which pops up a pointless box at the start to let you know the application has started.

```
from guizero import App

app = App(title="pointless pop-ups")

app.info("Application started", "Well done you started the
application")

app.display()
```

Running your application, you will see that an 'info' box appears (**Figure 3**). The first parameter passed to info is the title of the window; the second parameter is the message.

You can change the style of this simple pop-up by using `warn` or `error` instead of `info`.

Pop-up boxes can also be used to get information from the user. The simplest is a `yesno` which will ask the user a question and get a True or False response. This is useful if you want a user to confirm before doing something, such as deleting a file. Perhaps not every time they press a button, though!

Import the PushButton widget into your application:

```
from guizero import App, PushButton
```

Create a function which uses the `yesno` pop-up to ask for confirmation.

```
def are_you_sure():
    if app.yesno("Confirmation", "Are you sure?"):
        app.info("Thanks", "Button pressed")
    else:
        app.error("Ok", "Cancelling")
```

Add the button to your GUI which calls the function when it is pressed.

```
button = PushButton(app, command=are_you_sure)
```

Your code should now resemble **05-worlds-worst-gui.py**. When you run the application and press the button, you will see a pop-up asking to you confirm with a Yes or No (**Figure 4**).

You can find out more about the pop-up boxes in guizero at **lawsie.github.io/guizero/alerts**.

How about combining all of these 'features' into one great GUI?

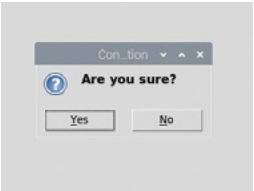


Figure 4 Yes, we're sure!

WINDOW WIDGET

Pop-up boxes can be used to ask users questions, but they are really simple.

If you want to do show additional information or ask for supplementary data, you could use the Window widget to create multiple windows.

Window is used in a similar way to App and has many of the same functions.

```
from guizero import App, Window

app = App("Main window")
window = Window(app, "2nd Window")

app.display()
```

You can control whether a Window is on screen using the `show()` and `hide()` methods.

```
window.show()
window.hide()
```

An app can be made to wait for a window to be closed after it has been shown, by passing True to the `wait` parameter of `show`. For example:

```
window.show(wait=True)
```

You can find out more about how to use multiple windows in the guizero documentation: **lawsie.github.io/guizero/multiple_windows**.

worst1.py / Python 3

DOWNLOAD
magpi.cc/guizerocode

```
# Imports ---------------

from guizero import App, Text


# App -------------------

app = App("its all gone wrong", bg="dark green")

title = Text(app, text="Hard to read", size="14", font="Comic
Sans", color="green")

app.display()
```

worst2.py / Python 3

```
# Imports ---------------

from guizero import App, Text


# Functions -------------

def flash_text():
    if title.visible:
        title.hide()
    else:
        title.show()


# App -------------------

app = App("its all gone wrong", bg="dark green")

title = Text(app, text="Hard to read", size="14", font="Comic
Sans", color="green")

app.repeat(1000, flash_text)

app.display()
```

worst3.py / Python 3

```
# Imports ---------------

from guizero import App, Slider, Text
from time import ctime


# Functions -------------

def update_date():
    the_date.value = ctime(date_slider.value)


# App -------------------

app = App("Set the date with the slider")
the_date = Text(app)
date_slider = Slider(app, start=0, end=999999999, command=update_
date)

app.display()
```

worst4.py / Python 3

```
# Imports ---------------
from guizero import App, Combo
from string import ascii_letters


# App -------------------

app = App("Enter your name")

a_letter = Combo(app, options=" " + ascii_letters, align="left")

app.display()
```

worst5.py / Python 3

```
# Imports ---------------

from guizero import App, Combo
from string import ascii_letters


# App -------------------

app = App("Enter your name")

name_letters = []
for count in range(10):
    a_letter = Combo(app, options=" " + ascii_letters,
align="left")
    name_letters.append(a_letter)

app.display()
```

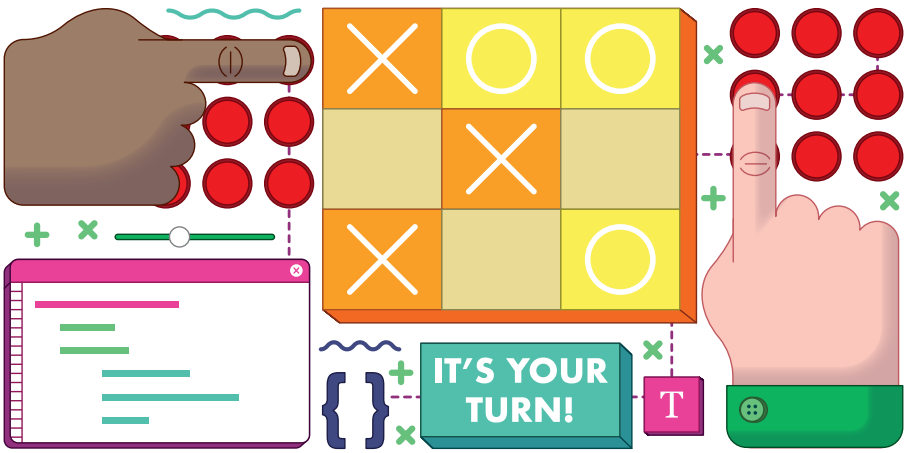
05-worlds-worst-gui.py / Python 3

```
from guizero import App, PushButton

def are_you_sure():
    if app.yesno("Confirmation", "Are you sure?"):
        app.info("Thanks", "Button pressed")
    else:
        app.error("Ok", "Cancelling")

app = App(title="pointless pop-ups")

button = PushButton(app, command=are_you_sure)

app.info("Application started", "Well done you started the
application")

app.display()
```

Chapter 6

Tic-tac-toe

Use your GUI to control a simple game



Now that you have learnt how to make a basic GUI, let's add some more programming logic behind the scenes to make your GUI work as the means of controlling a game of tic-tac-toe (also known as noughts and crosses).

Create a new file with the following code:

```
# Imports ---------------
from guizero import App

# Functions -------------

# Variables -------------

# App -------------------
app = App("Tic tac toe")

app.display()
```