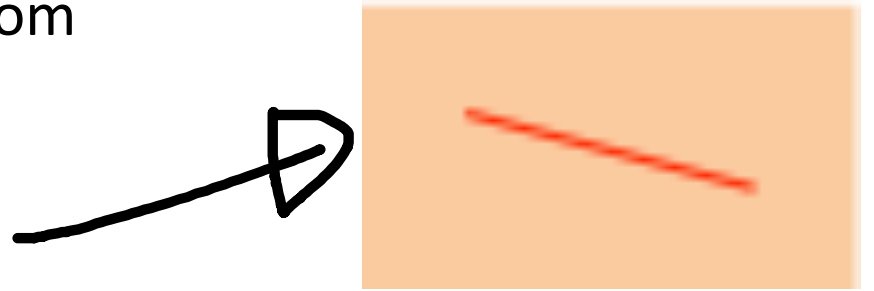

Graphical User Interfaces I
Week 5 Presentation 4

Graphical User Interface Programming

- Event driven programming is a different paradigm for creating a sequence of instructions and possible interactions with a user
- Event handlers / ”callbacks” are functions associated with certain events
- Different from other types of programming you are familiar with
- You rely on packages which you must import and use in your own program (there are many options)
- Need to interact closely with an environment (operating system, windows, display, etc)

Many Different Packages

- First of all you don't code everything by yourself, and instead use functions from packages that you must import when writing your program.
- There are **low-leve**l packages with functions (called "primitives") for performing tasks such as drawing a rectangle, a line or a curve or 3d rendering.
- You also have **high-level** packages that use the previous ones to draw for instance buttons, and automatically change them when they are clicked.



Historical Overview of Java GUI Packages

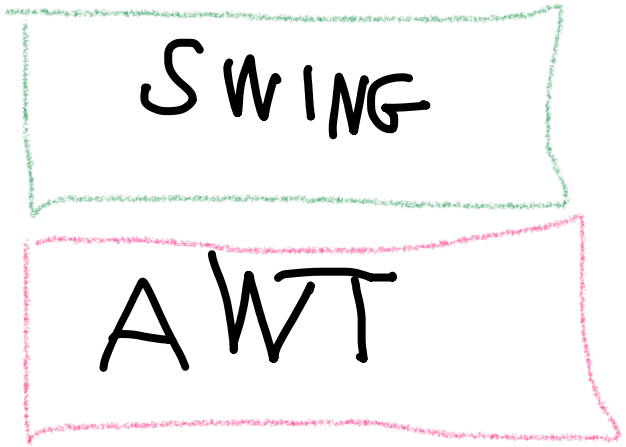
- 1995 Abstract Windows Toolkit (‘**AWT**’)
  Looks like other applications on the system

- Dec 1996 Java Foundation Classes (‘**Swing**’)
  Looks the same on all systems

In Java, several packages allow you to code a GUI. The first one was AWT, followed by "Java Foundation Classes" quiclky renamed "Swing".

Historical Overview of Java GUI Packages

```
import java.awt.*;
import javax.swing.*;
import javax.imageio.*;
```

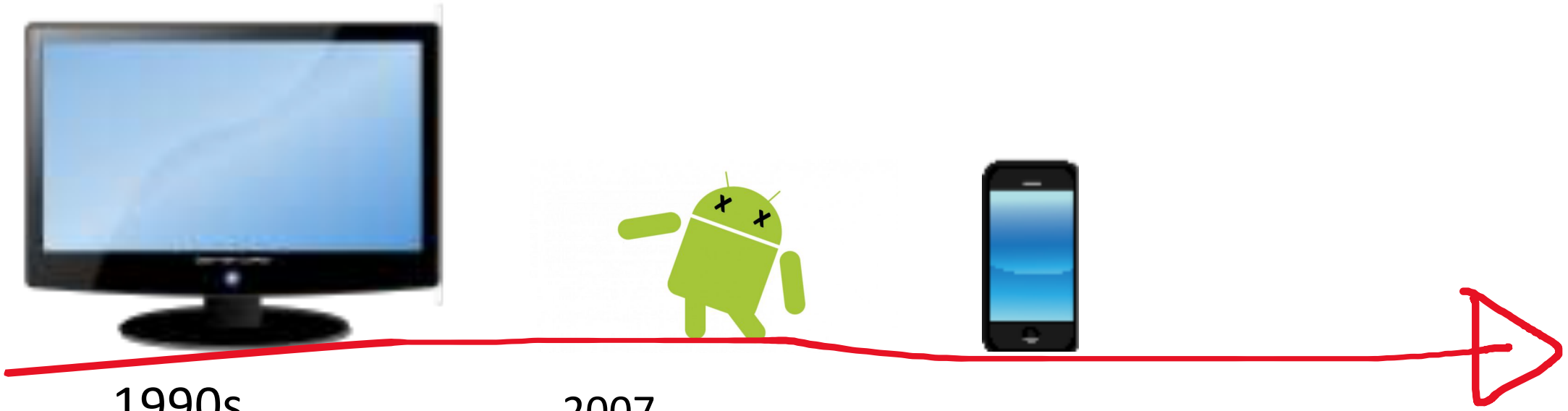


Swing relies on AWT, and whenever you code a Swing application you also need to import classes from AWT, as well as from other packages for images.

Historical Overview of Java GUI Packages

- A new package, JavaFX, was introduced in 2008.

2008 JavaFX `import javafx.*;`



1990s

2007
Android officially adopted by many Smartphones

JavaFX

JavaFX, with which you import classes from a single package (but many subpackages) supports other devices than computer screens for which AWT and Swing were written – mobile phones in particular. It also allows to define the looks of applications in external files called "style sheets" or "CSS" files (CSS means "Cascading Style Sheet" – 'cascade' is French for 'Waterfall'), a technique borrowed from web programming. However, because software has a long life, there is a lot of Swing around, Swing is still much in use and will probably stay around for quite a while. It's good to know both Swing and JavaFX (they aren't VERY different, class names change, basic ideas are the same).

Swing and AWT are replaced by the JavaFX platform for developing rich GUI applications.

The Basic Structure of a JavaFX Program

The `javafx.application.Application` class defines the essential framework for writing JavaFX programs.

Historical Overview of Java GUI Packages

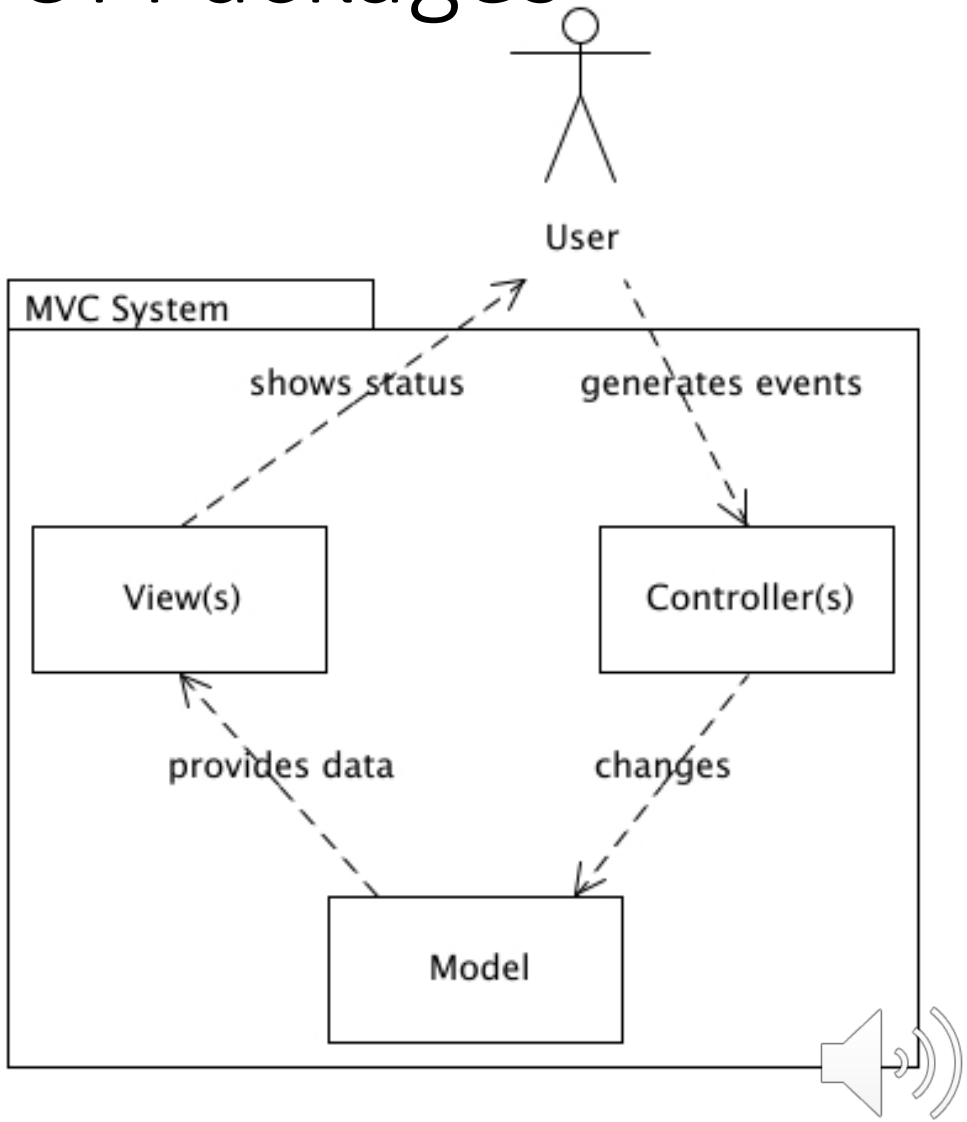
2008 JavaFX import javafx.*;

Model = data management

view = user interface (visual elements/looks)

Controller = Logic

JavaFx applications often follow a popular structure known as "Model/View/Controller" (or MVC) in which data management, user interface and logic are clearly separated.



Event Driven Progamming

Whichever package you are using, and even whichever programming language you are using (what I'm saying about Java is also true in C/C++ or Python for example), programming graphical interfaces is a very different kind of programming that what you have done so far, and is called event- driven programming.

A Graphical Application is just a big loop

You don't have to code the loop, it's performed for you by the graphical package functions. Basically, you draw things on the screen, display them, and run a loop that does nothing but wait. What is it waiting for? Simply for the user who (presumably) is sitting in front of the screen to do something (other than head-scratching).

WAIT

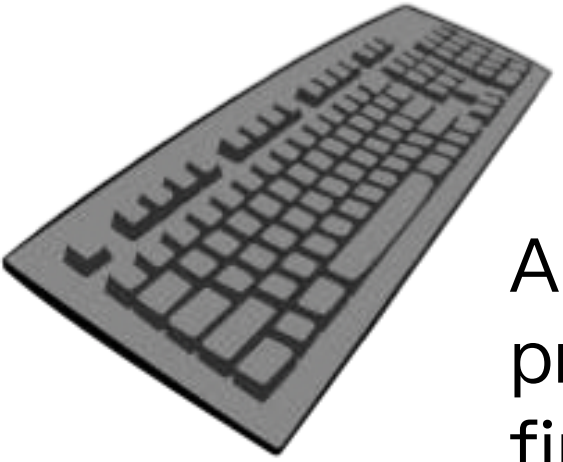
Your application sleeps and waits.

EVENT

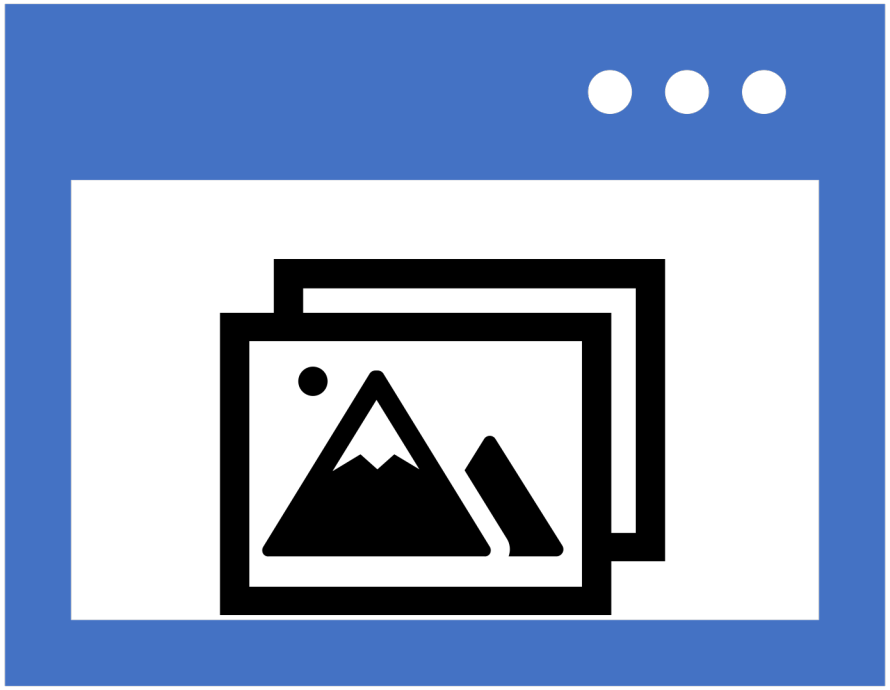
Until the user does something (the picture is an allegory).

Reaction

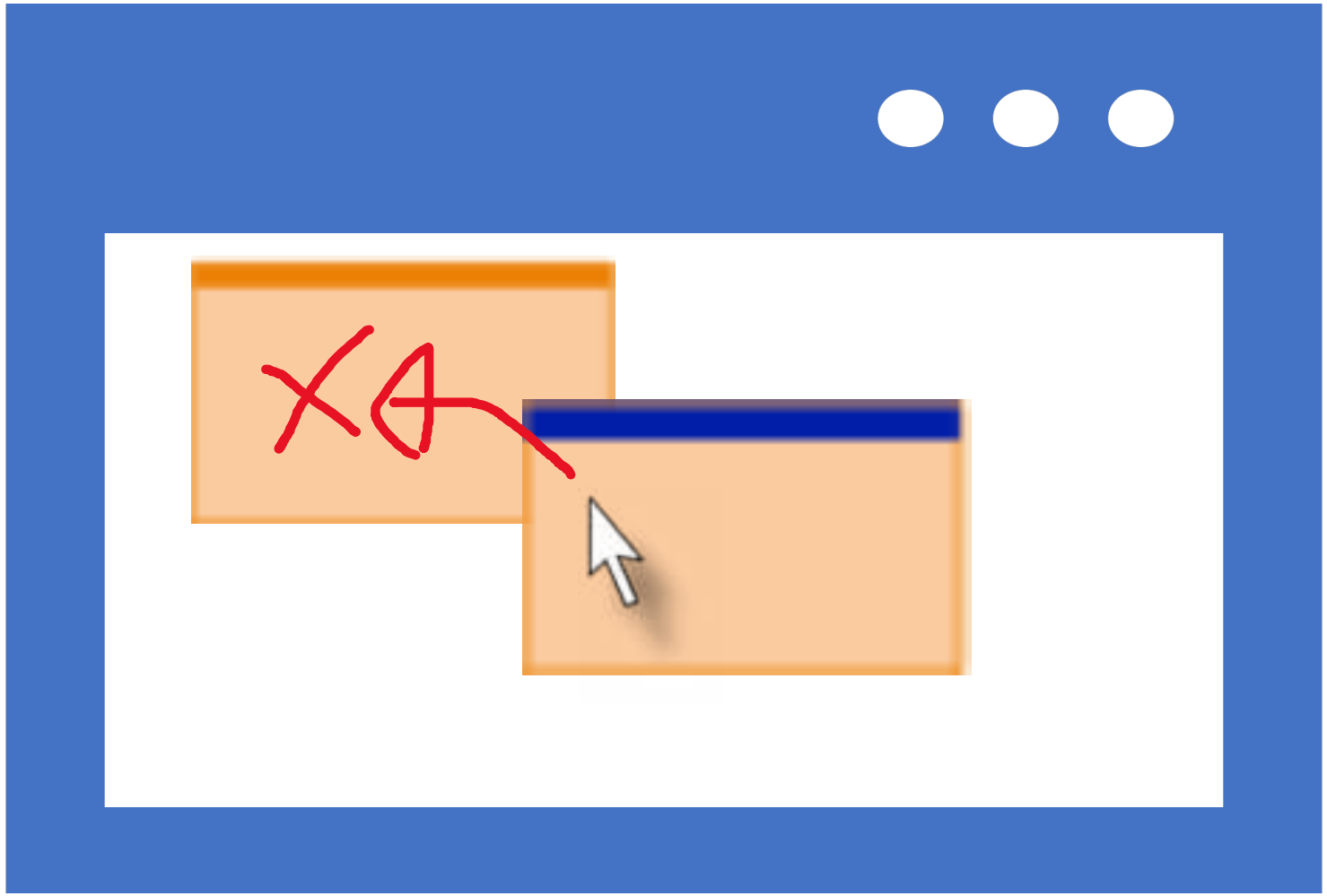
At which point your program is expected to react and do something.



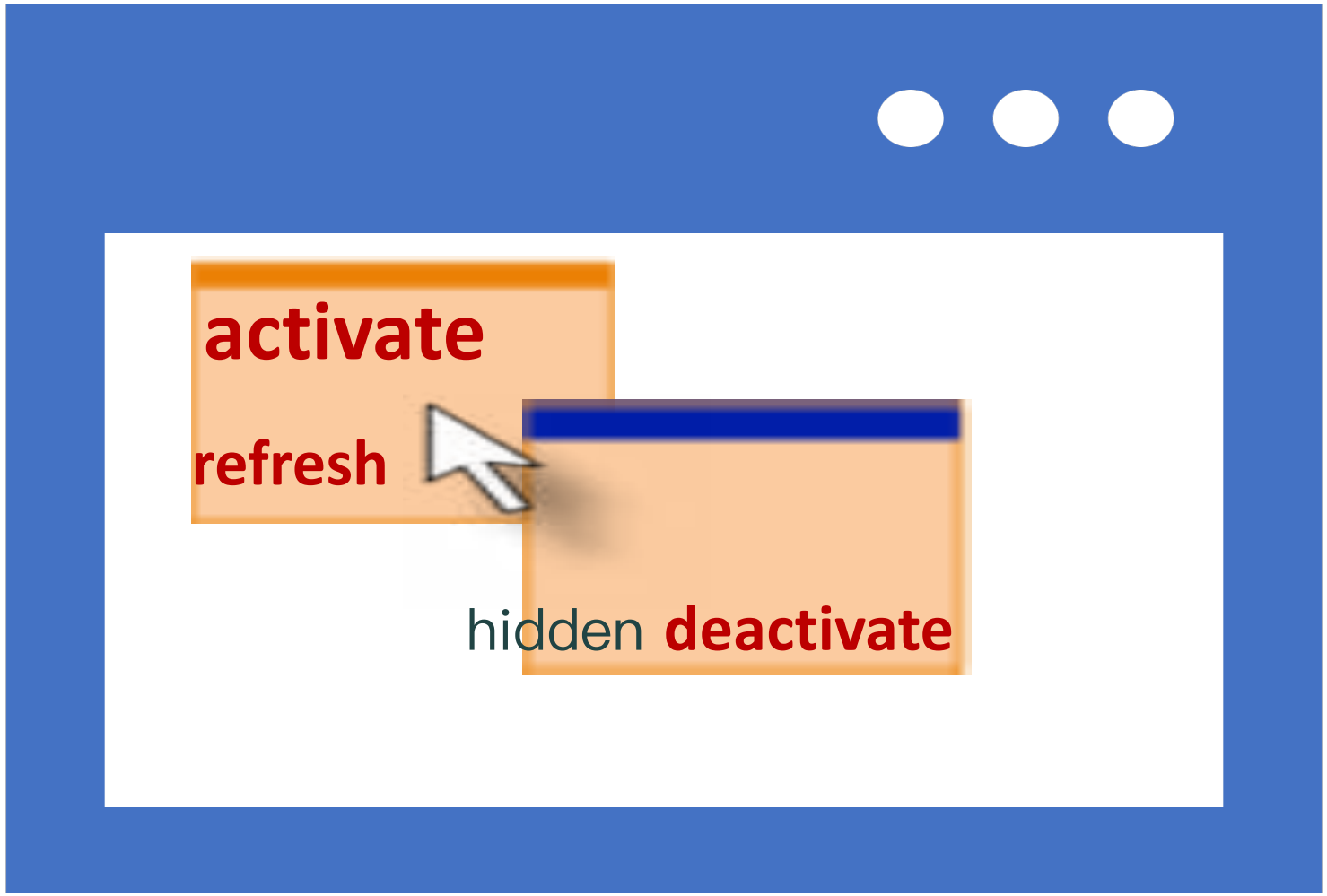
An event can be anything. A key pressed, a move of the mouse, a finger swiping a touch screen, somebody jumping in front of a webcam ... Anything that can be translated into an electrical signal reaching the computer.



Your graphical application will run in a Window Manager that also reacts to events. For instance you may have an active window and an inactive window on the screen.

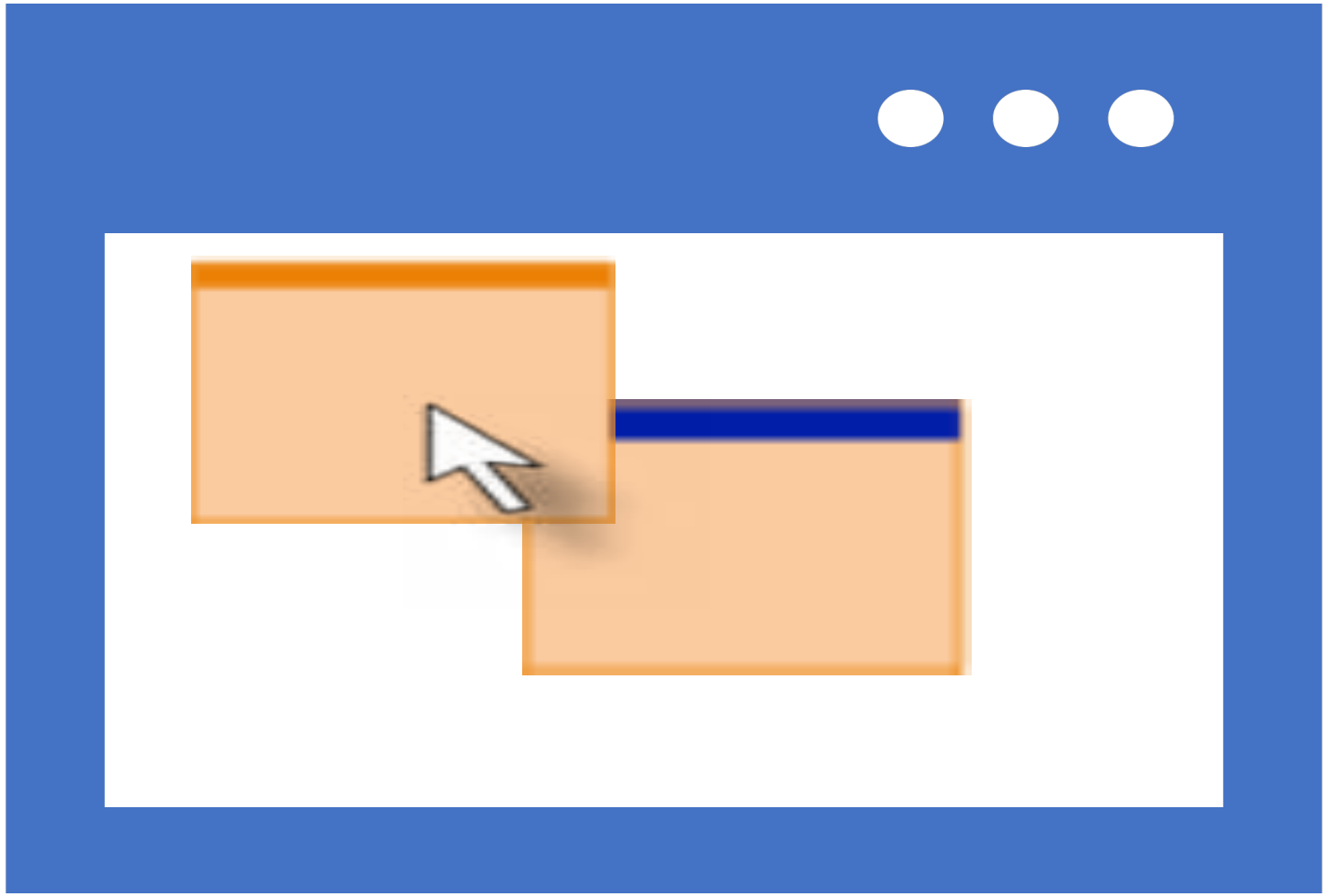


The user may move the mouse and click it outside the active window. The windows management system will get the coordinates and discover that another window is at this location. This is an event.



This means the inactive window has to be brought forwards. Note: there is a stack of objects to represent this internally.

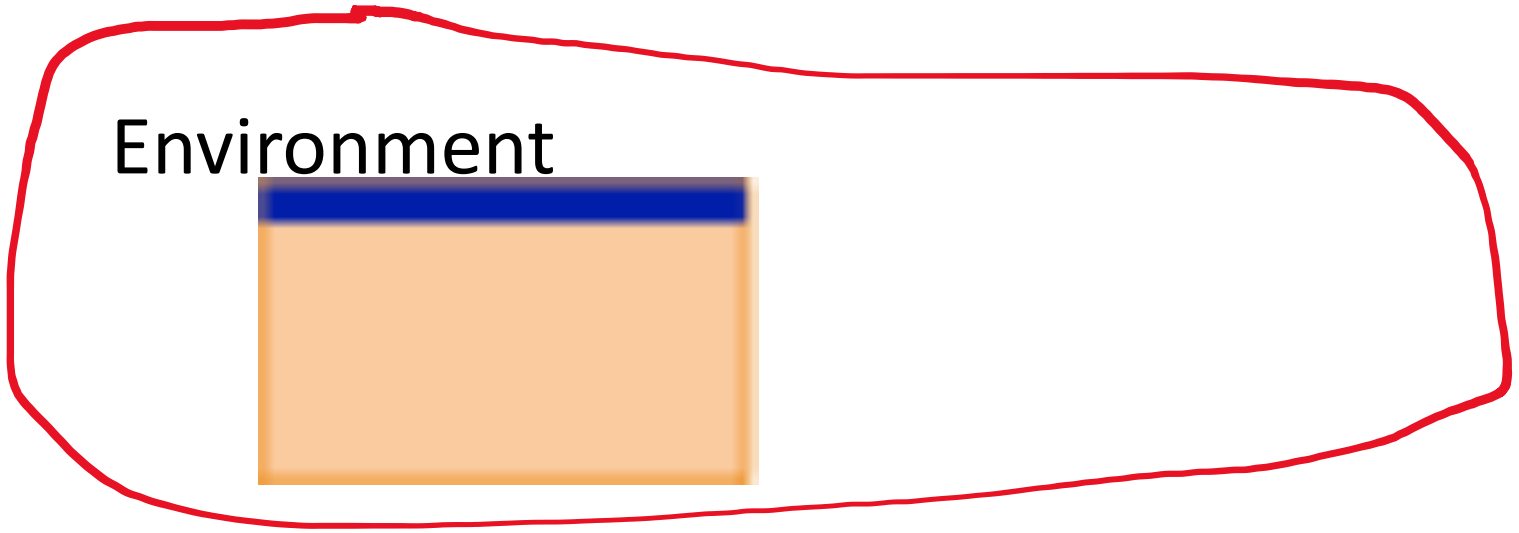
The previously active window must be redrawn to show it is inactive, the newly active window must be redrawn too. Including what was previously hidden.



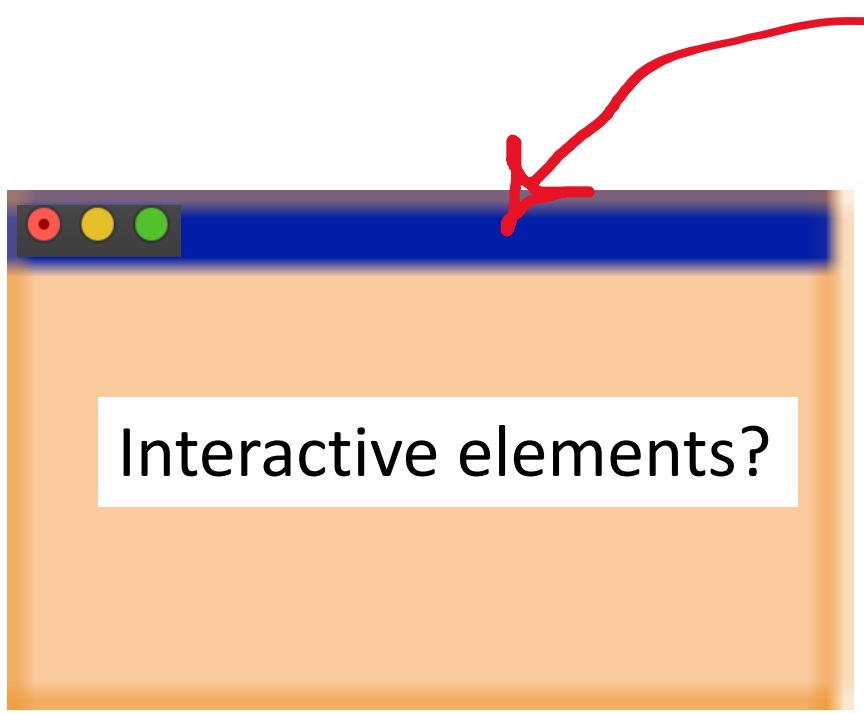
Then your screen will look like this. All this stuff is performed by the window manager and doesn’t require you to write any code.

Environment

Observe that the GUI is running within an environment that provides services for managing windows (resizing, destruction, etc).

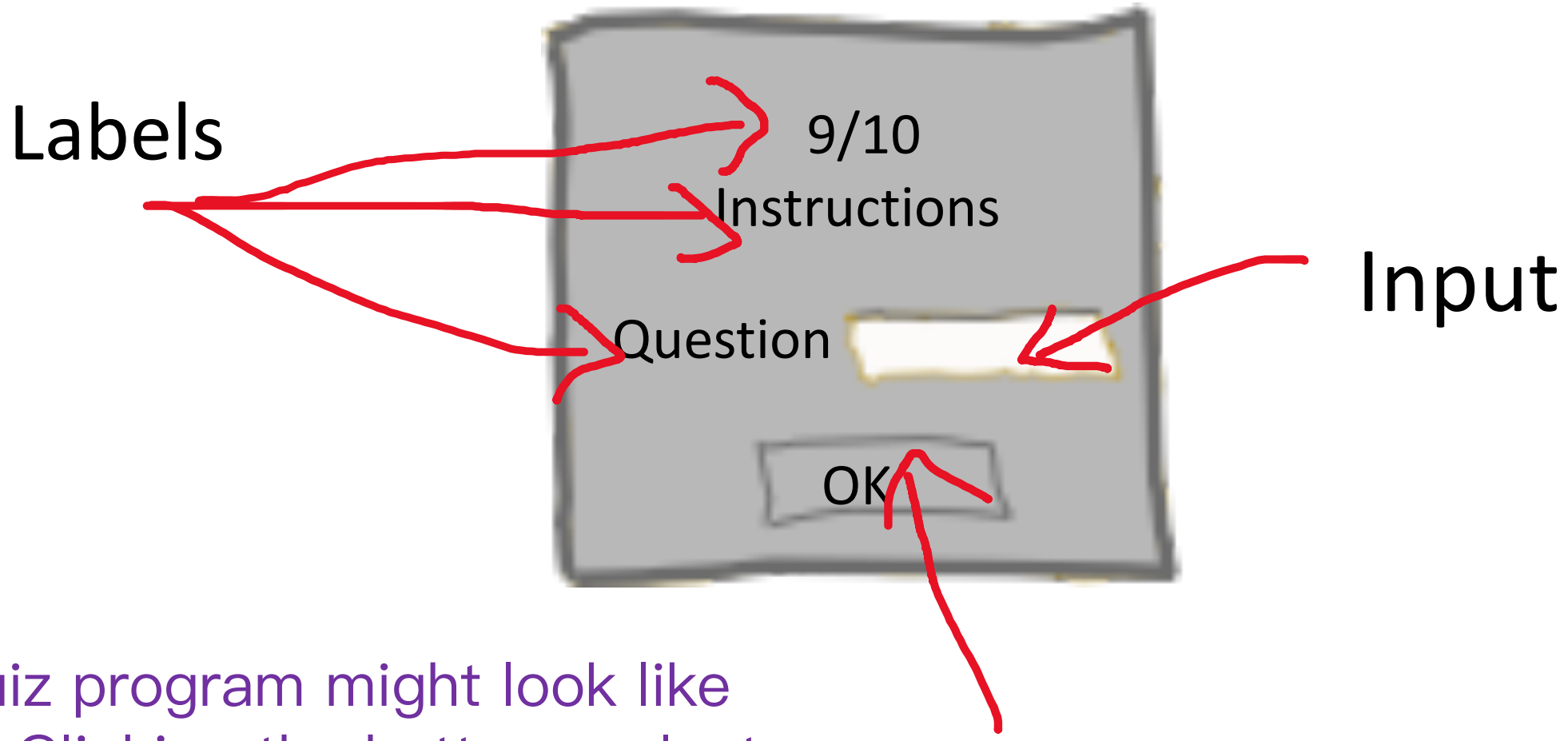


When you design your application you must decide on what the user will see: will your window have a title, will it be resizable, which elements will the user interact with in the window?



What does the user see?

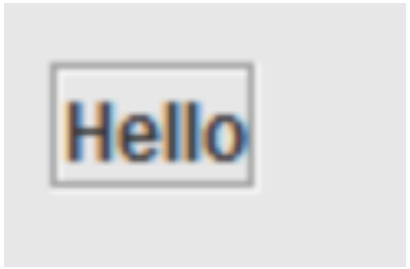
Components of a window...



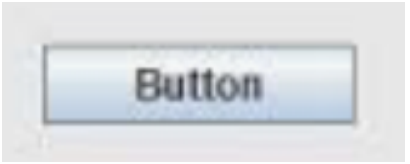
A quiz program might look like this. Clicking the button evaluates the answer.

Your program must prepare everything in advance and, like a conjuror, only reveal elements at the right time when they should be visible.

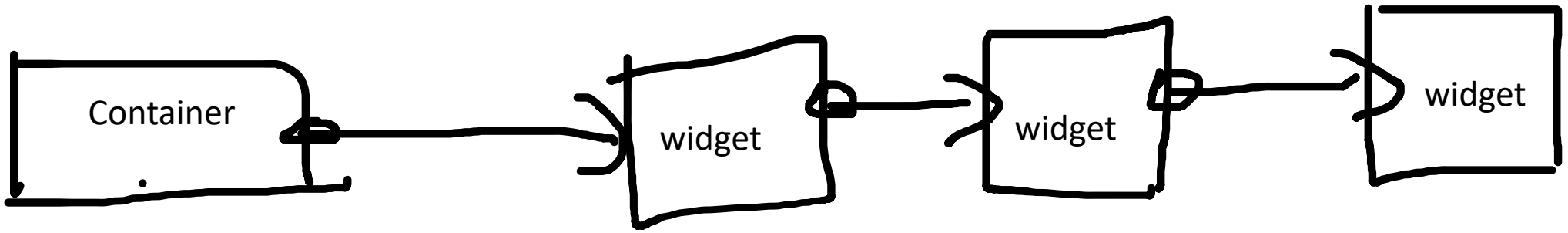
Widget = Window Gadget



- Visual elements are called “widgets” - short for window gadgets
- Some widgets include:
  - labels
  - entry fields
  - drop down lists
  - check boxes
  - and buttons



Containers



Something that you are probably not familiar with is the notion of "container". If you see widgets, you don't see containers that are nothing more than linked lists of widgets and (more about this soon) other containers.

Layout

- The purpose of containers is to make creating a layout easier.
- A layout means how the various widgets are displayed on the screen in relation to each other.

If you have a fixed–size window, things are easy. You can say "I want this widget to appear at these coordinates relative to the upper–left corner of the window".

Unfortunately the easy case isn't the most common…

Some containers are fixed

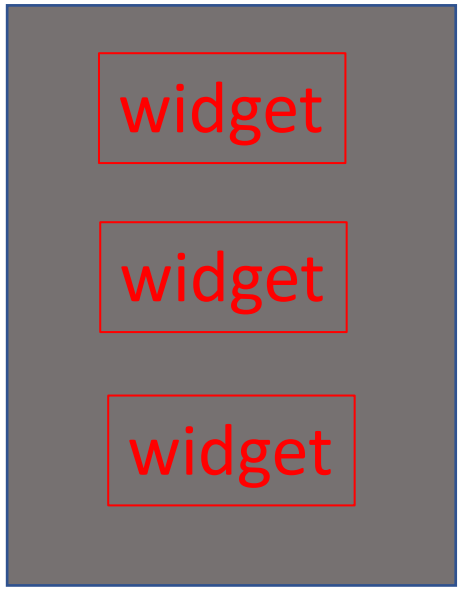
Usually people can and do resize windows and you want a “fluid layout”. If you gave absolute coordinates assuming a given window size it will soon be a big mess.

Most are not

Boxes

Containers are for solving these issues. Boxes come in two flavours and display widgets next to each other (with some padding in between) in only one direction.

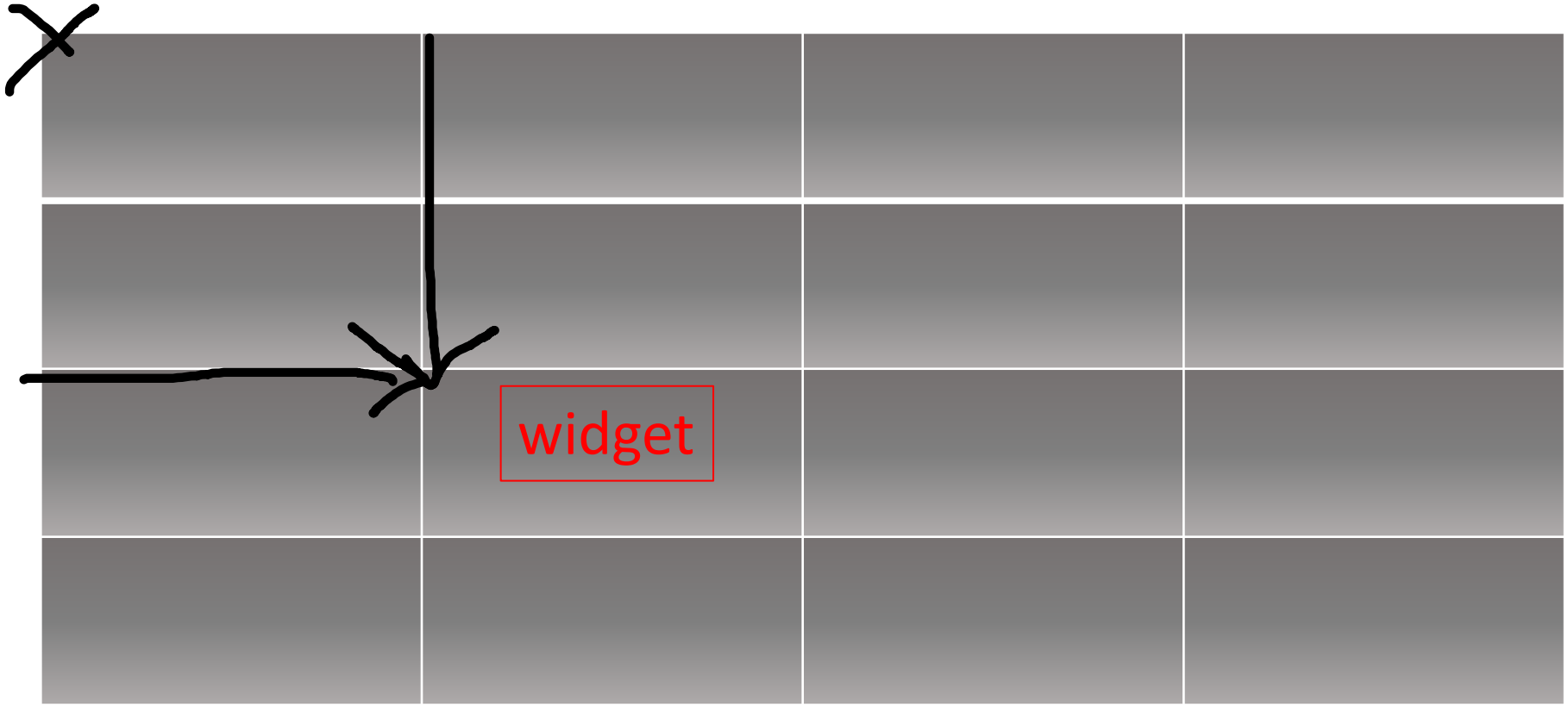
Horizontal Boxes



Vertical Boxes

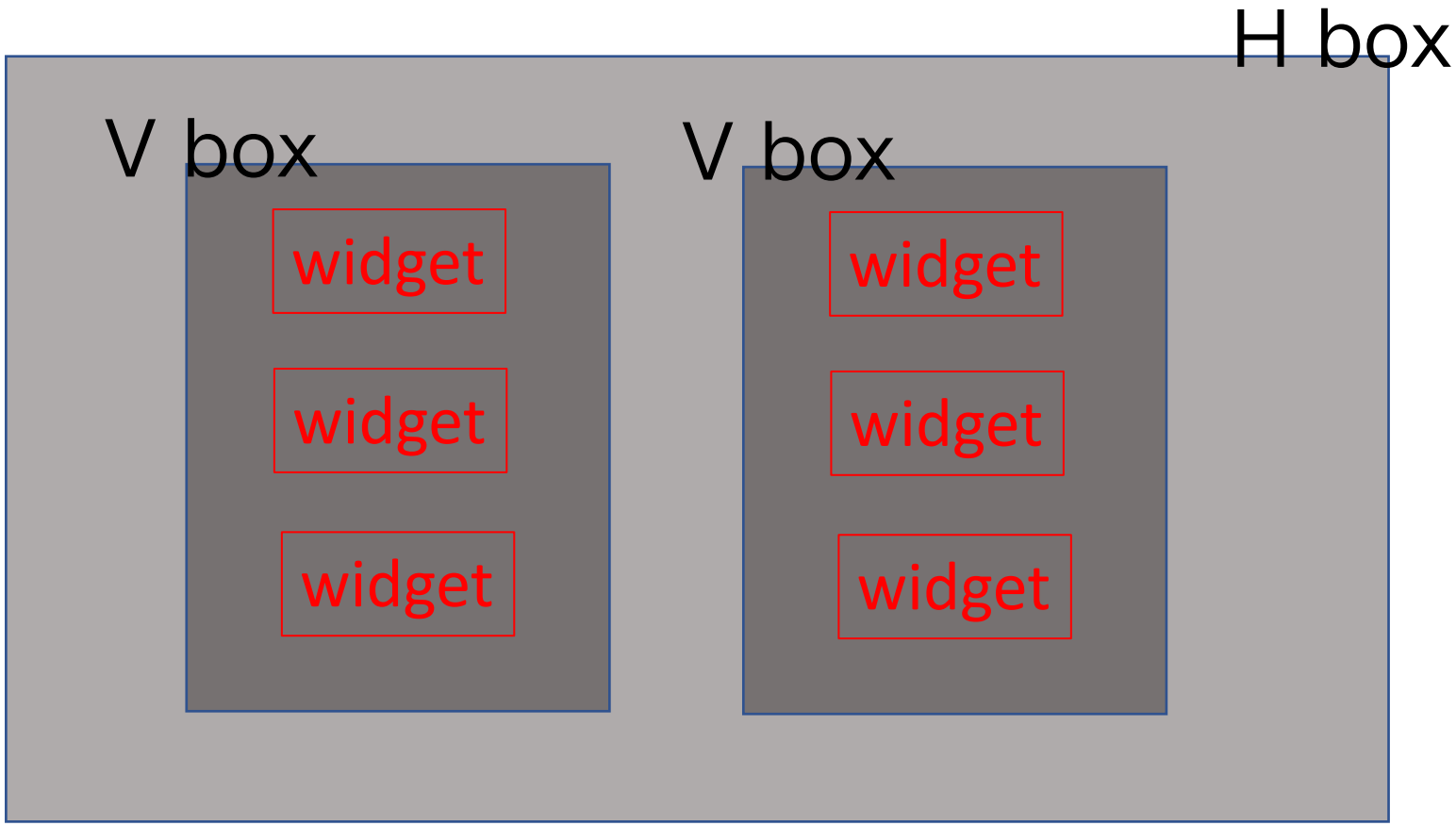
Grids

Other frequently used containers are grids which allow you to place widgets at **relative** row column coordinates instead of absolute distance coordinates



The great thing with containers is you can have nested containers.

For instance you can have two vertical boxes (each one showing widgets vertically) and add them to a horizontal box (side by side). When the window is resized, the global layout is respected and it still looks (more or less) as intended



Callback

- The last important idea to understand with graphical user interfaces is the one of "callbacks", often called "handlers" in Java, which is the name given to a function associated with an event. For instance, clicking a button might trigger a search inside a database. This is a function that you write, and associate with the button.

- A “callback” is a function associated with an event

Predefined Events

There is an extensive set of predefined events. You only handle those that matter to you. You must often perform a number of checks when the window is destroyed. For instance a text editor will ask you whether you want to save your changes.

Destroy window

Button press / release

Key press / release

Focus in / out

Move in / out