Graphical User Interfaces

Tkinter

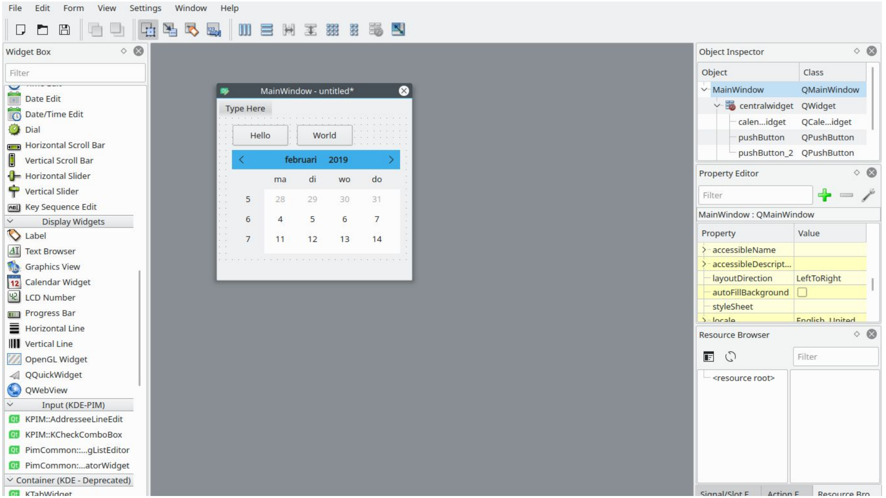
Why Tkinter?

- Comes with Python
- Cross platform
- Mature and stable (over 30 yrs old!)
- Sufficient for small, lightweight GUIs
- Offers a glimpse at core principles of a GUIs

Tkinter

Why Not Tkinter?

- Lacks modern widgets (components)
- Does not have a GUI designer
- Slower than other options
- Does not support accessibility
    - Some third party support (Tka11y) but not cross-platform (no Windows/OSX)



Python GUI designer in QT

Tkinter

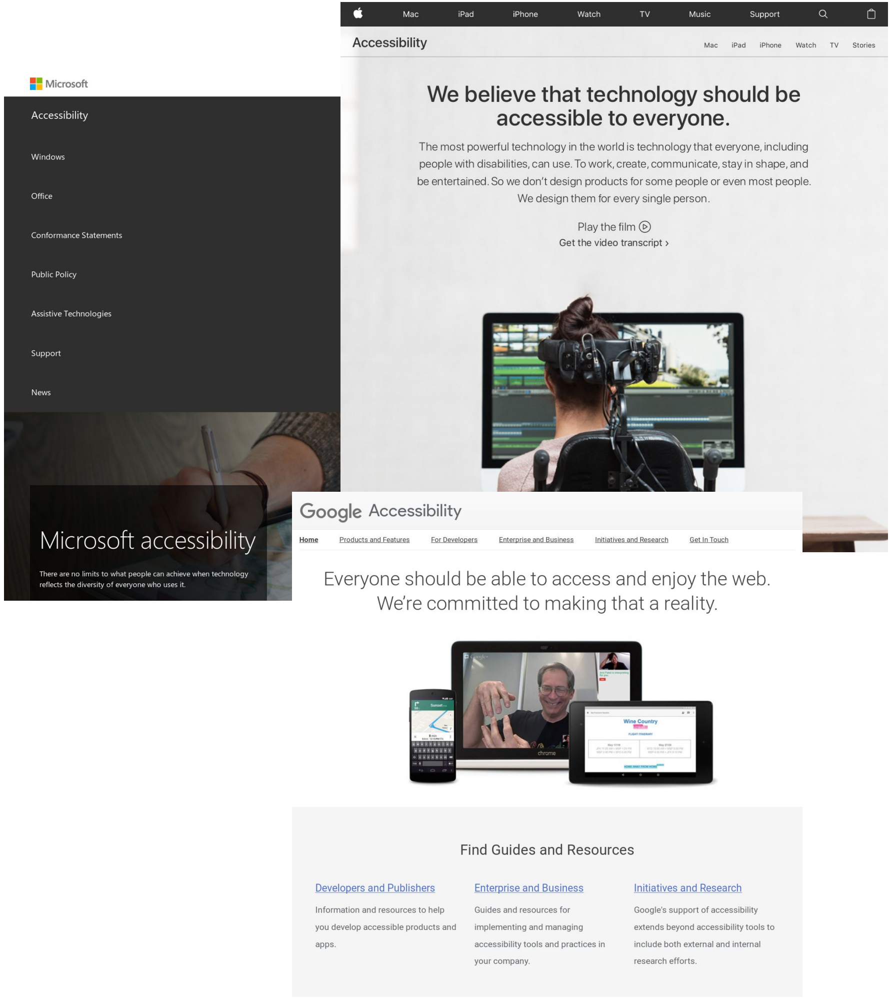
Why accessibility?

- Any program designed to be used by the population at large should be accessible.
- In GUI design, accessibility means that the accessibility layer of the OS can interpret and act upon GUI widgets without user intervention.

Tkinter

Why accessibility?

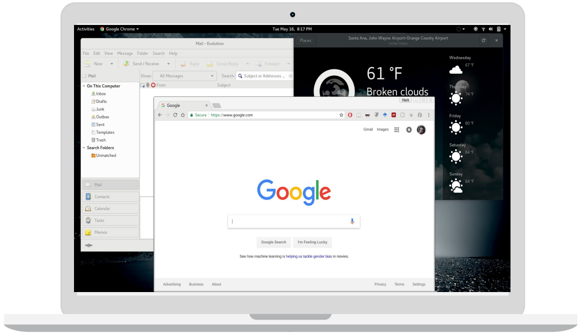
- All major software producers have accessibility initiatives and look for developers with accessibility experience.
- But also, many software producers do not. THEY NEED YOUR HELP.



Tkinter

What is accessibility?

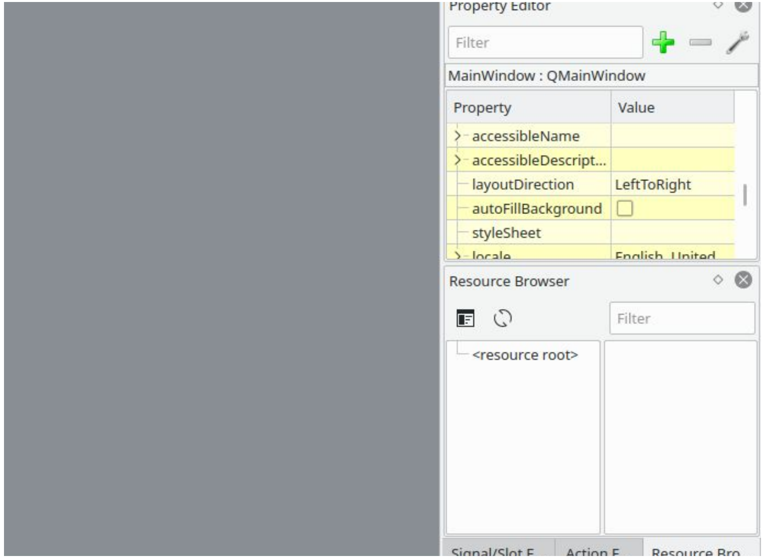
- Ensuring all people, regardless of ability, can access and interact with graphical user interfaces.
- Includes: visual impairments, auditory impairments, cognitive and motor impairments.



Tkinter

How are GUI’s made accessible?

- Support keyboard only
- Consistent keyboard conventions (e.g., ctrl+c == copy to clipboard)
- Described control and actions
- Avoid communicating through images. If necessary, provide alternative text descriptions.



Python GUI designer in QT

Designing a GUI

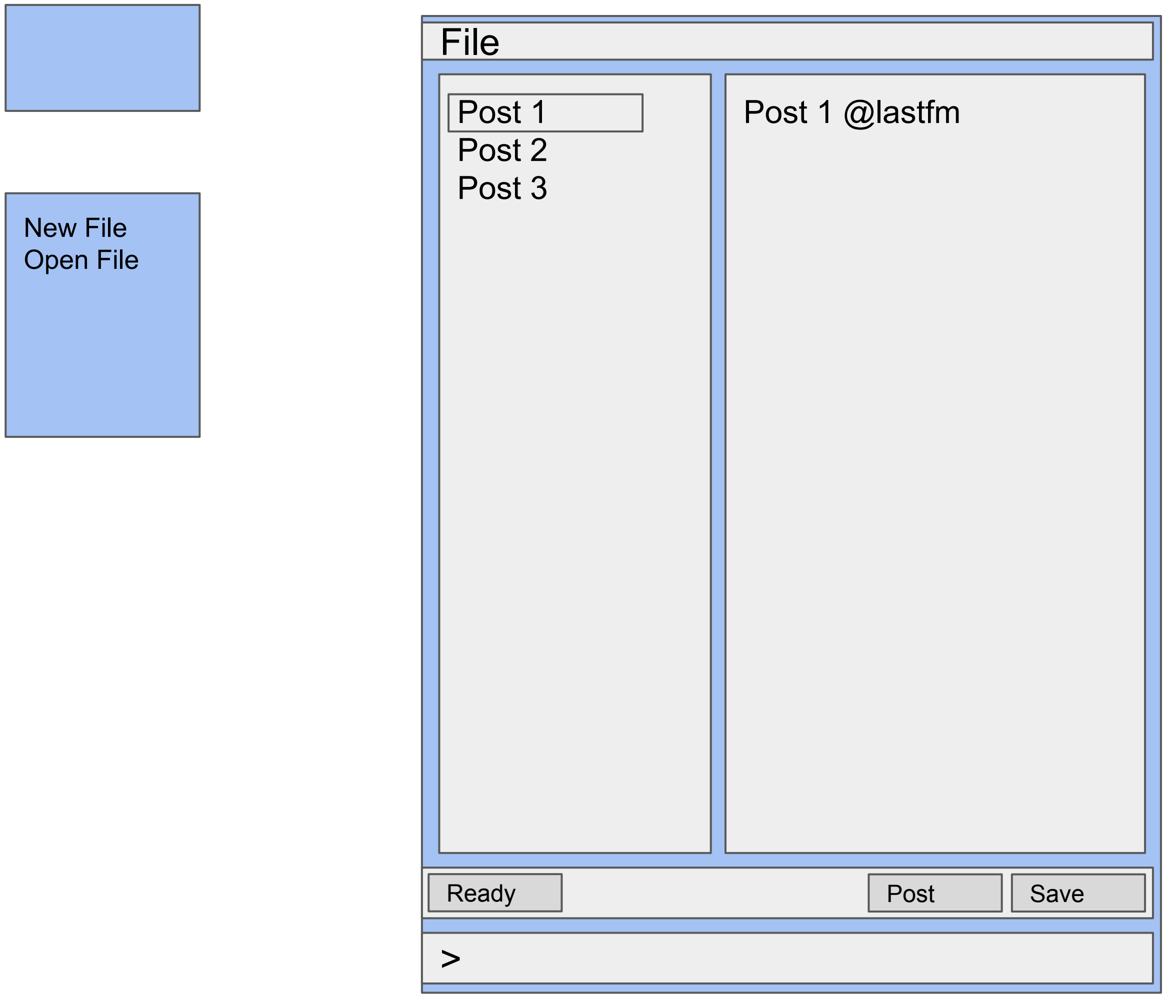
File

Ready

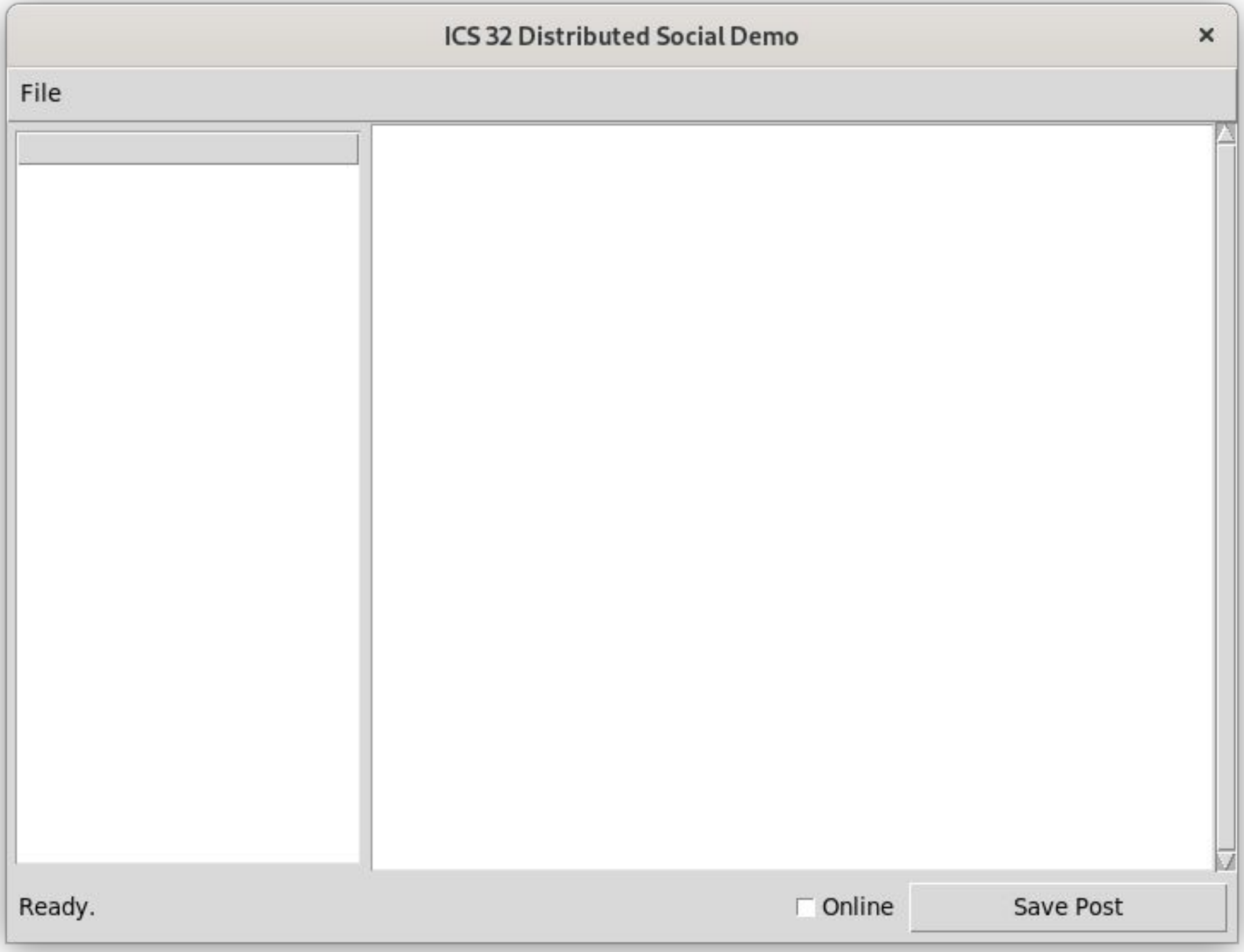
Post

Save

New File
Open File


New File
Open File
File
Post 1
Post 2
Post 3
Post 1 @lastfm
Ready
Post
Save
>

Geometry Managers


ICS 32 Distributed Social Demo
File
Ready.
Online
Save Post

Geometry Manager

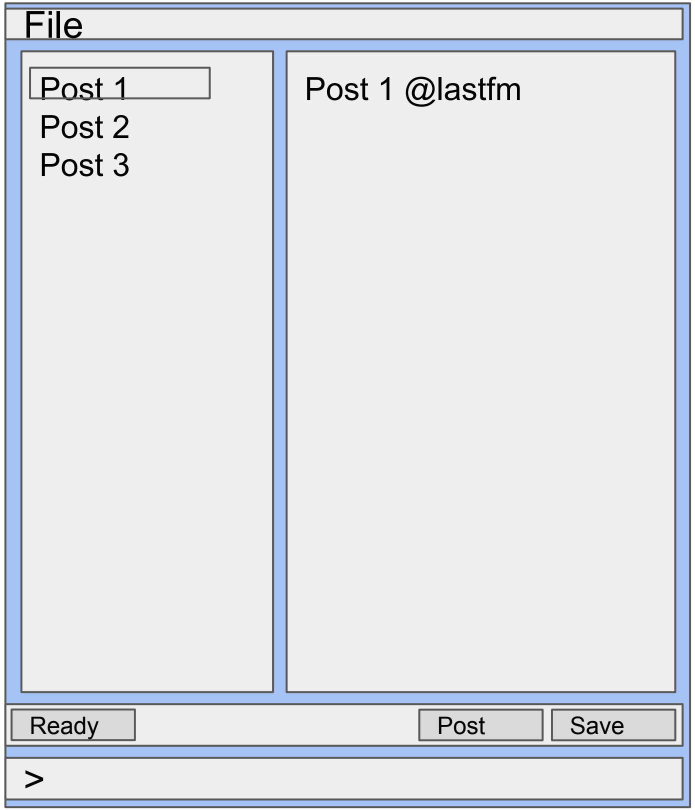
- A set of algorithms that determine how to render widgets in a GUI from configuration parameters.
- Config parameters are set programmatically, by YOU.
- Different types of algorithms = different types of geometry managers

Tkinter Geometry Managers:

- **Pack**
- Place
- Grid

Geometry Manager: Pack

- Determines a ‘parcel’ or rectangular space large enough to hold the specified widgets (e.g., button, textbox, checkbox)
- Responsive, works well across platforms
- By default will center the widget(s)
- But! Widgets can be assigned rules to further control where they are placed in the parcel:
    - Fill - specify which direction (horizontal, vertical, both)
    - Side - specify which side (top, bottom, left, right)
    - Expand -
- Packing order is important!
- Let’s play…



Geometry Manager: Place

- Easy to conceptualize, difficult to get right for cross-platform, varied resolutions. Not responsive.
- Widgets are placed within a frame according to specific x,y coordinates
- Good for windows that will always be fixed in dimension:
    - Pop-up dialog
    - Simple input (a textbox and button)

```
btn = tk.Button(...)

btn.place(x=10,y=100)
```

Geometry Manager: Grid

- Benefits of both pack and place. Responsive, easier to understand and design.
- Widgets are placed within a grid of rows and columns
- Rows and columns can be customized with attributes (borders, padding, min/max size, ‘stickiness’)
- A bit more complex, programmatically, but probably best option once understood.

```
btn = tk.Button(...)

btn.grid(row=0, column=3,
sticky="nsew")
```

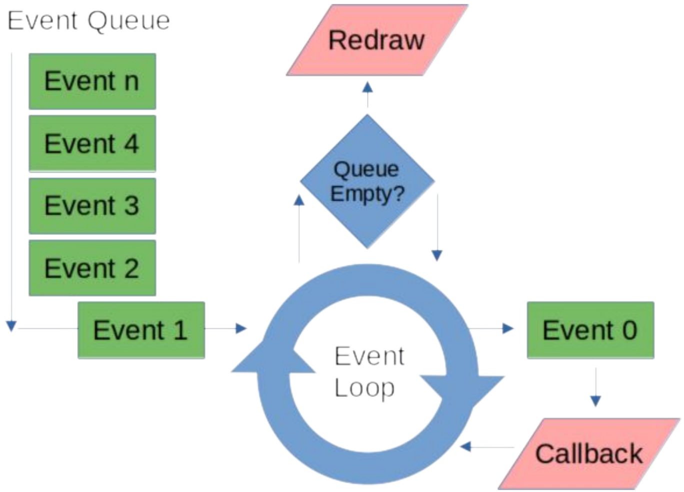
Geometry Managers

You don’t have to pick just one! Pick the one that is right for your layout. Mix and match as needed.

The Event Loop

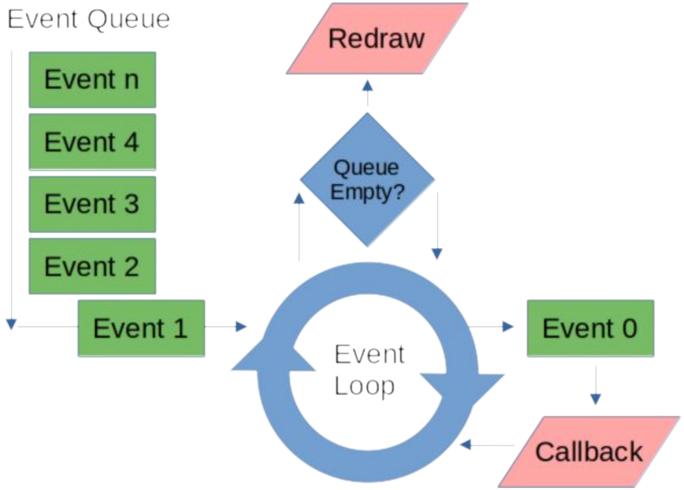
The Event Loop

- In graphical interface programming, screen updates are processed in an event loop.
- As each graphical element (widget) is acted upon, it is put into an event queue, and must wait its turn to be executed.



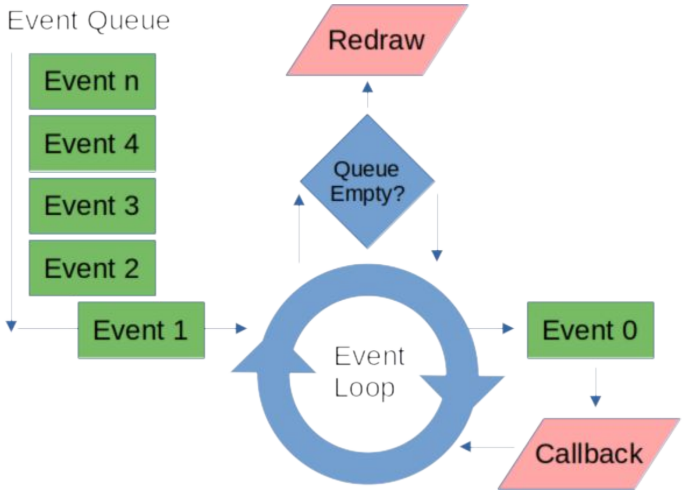
The Event Loop

- Once all events in the queue are processed, the event loop initiates a redraw or updates the screen.
- These events often go unnoticed because they occur so quickly.
- Until some other process takes a long time.



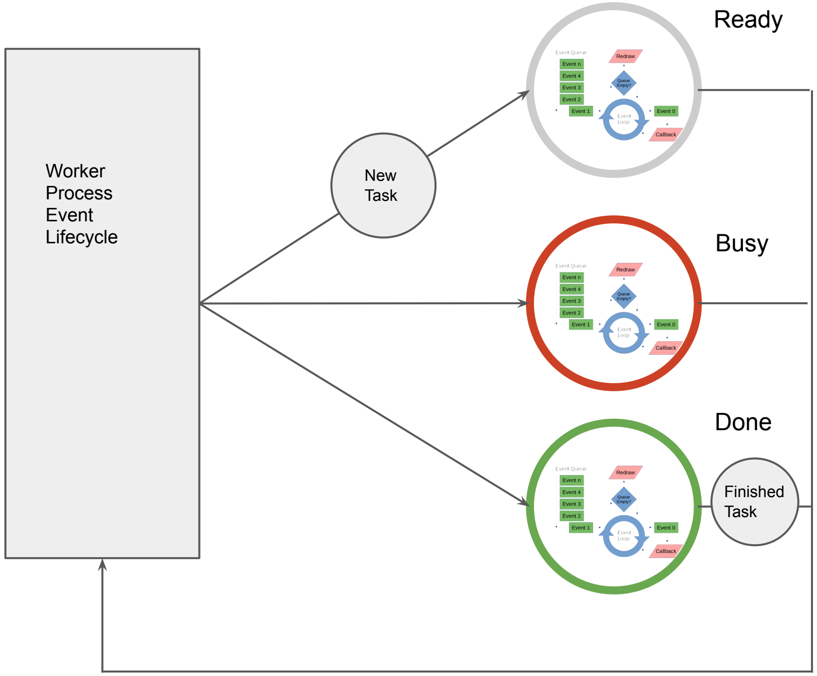
The Event Loop

- Demo 1



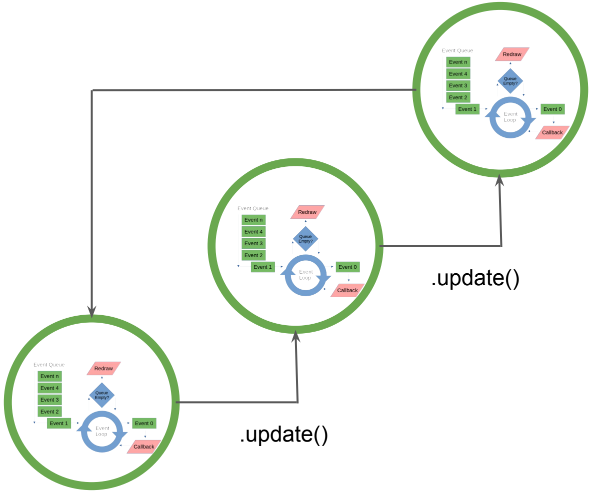
The Event Loop

- Wait...What's a thread!?
  - Threads enable multiple processes to run at the same time.
  - Get complicated quickly.
  - Require special coding to share information
  - Way beyond the scope of ICS 32!



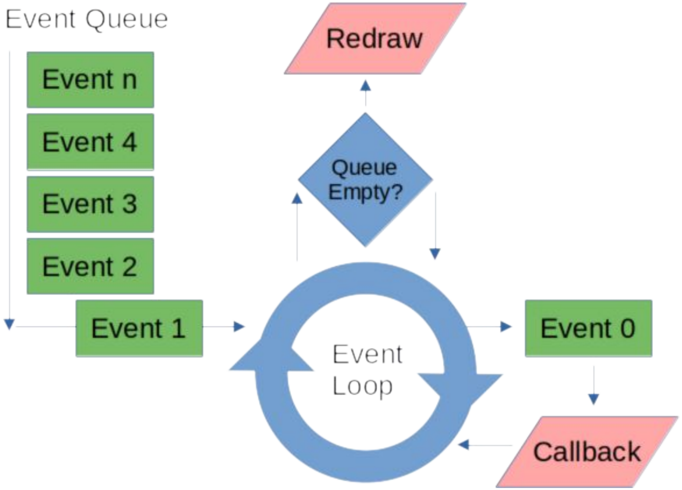
The Event Loop

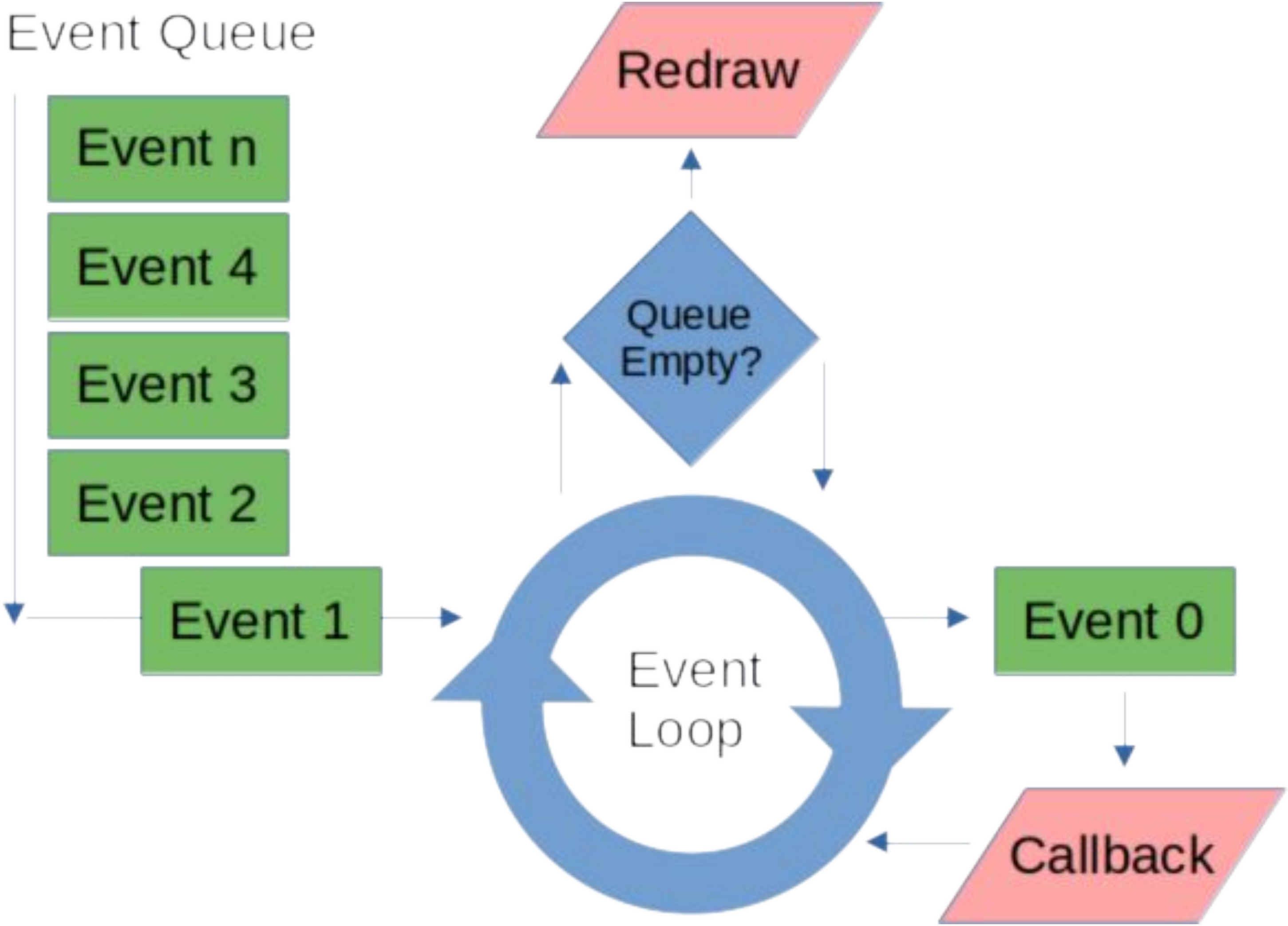
- A simple fix, suitable for ICS 32.
  - update() and update_idletasks()
    - Start a new event loop, nested within the existing one.
    - Forces event processing by starting a new event loop.
    - Update_idletasks is the same as update, but only processes screen redrawing, not other events in queue.
  - USE SPARINGLY! Nested event loops can quickly grow out of control and render unexpected results in your program.



The Event Loop

- Demo 2




Event Queue
Event n
Event 4
Event 3
Event 2
Event 1
Redraw
Queue Empty?
Event Loop
Event 0
Callback