Overview

This lesson explains what an event queue is in Tkinter and how the event loop uses it to decide what to do next.

Important Information

What an "Event" Is

An **event** is anything the program needs to respond to, such as:

- a mouse click
- a key press
- moving the mouse
- resizing a window
- a timer event created by `after`

Each event is like a short message that says, "Something happened."

What the Event Queue Is

The **event queue** is a waiting line of events. As events happen, they get placed into the queue. The event loop repeatedly takes the next event from the front of that line and processes it.

This is why GUIs feel responsive: events are handled in order, many times per second.

Events Are Processed in Order

If five clicks happen quickly, they will enter the queue and be processed one by one. If the event loop is running normally, the queue stays small because events are handled quickly.

If the event loop is blocked, the queue can fill up, but nothing gets handled until the blocking code finishes. That is when the program feels frozen.

Timer Events Also Go Into the Queue

When you use `after`, Tkinter schedules a timer event for the future. When that time arrives, the timer event enters the queue like any other event, and the event loop processes it.

```
from tkinter import *

root = Tk()

def say_hi():
    print("Hi from a timer event!")

root.after(1000, say_hi)  # about 1 second later, an event is queued

root.mainloop()
```

This is why `after` is so useful for long tasks. Instead of one huge callback, you schedule many small events that the event loop can handle between other user actions.

Why the Queue Explains "Lag"

If your event handlers are slow, the queue grows because events are arriving faster than they are being processed. That can cause delayed clicks, delayed key presses, and delayed screen updates. The fix is usually the same: make handlers faster, or break work into steps using `after`.