import threading  
import time  
import tkinter as tk  
from tkinter import ttk, messagebox  
from PIL import Image, ImageTk  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import matplotlib.pyplot as plt  
  
# -------------------- Account Class ---------------------  
class BankAccount:  
    def \_\_init\_\_(self, name, balance, priority):  
        [self.name](http://self.name/) = name  
        self.balance = balance  
        self.priority = priority  
        self.lock = threading.Lock()  
  
    def withdraw(self, amount):  
        self.balance -= amount  
  
    def deposit(self, amount):  
        self.balance += amount  
  
# -------------------- Transfer Animation ---------------------  
def animate\_transfer(label):  
    icons = [transfer\_icon1, transfer\_icon2, transfer\_icon3]  
    for \_ in range(3):  
        for icon in icons:  
            label.config(image=icon)  
            root.update\_idletasks()  
            time.sleep(0.3)  
    label.config(image=success\_icon)  
  
def draw\_arrow(canvas, nodes, from\_acc, to\_acc, amount):  
    from\_coords = nodes[[from\_acc.name](http://from_acc.name/)]  
    to\_coords = nodes[[to\_acc.name](http://to_acc.name/)]  
    arrow = canvas.create\_line(from\_coords[0], from\_coords[1], to\_coords[0], to\_coords[1],  
                               arrow=tk.LAST, width=3, fill="blue")  
    mid\_x = (from\_coords[0] + to\_coords[0]) / 2  
    mid\_y = (from\_coords[1] + to\_coords[1]) / 2  
    text = canvas.create\_text(mid\_x, mid\_y - 10, text=f"${amount}", font=("Arial", 12), fill="blue")  
    canvas.update()  
    time.sleep(2)  
    canvas.delete(arrow)  
    canvas.delete(text)  
  
# -------------------- Transaction Logic ---------------------  
eff\_data\_deadlock = []  
eff\_data\_nodl = []  
  
def transfer\_priority(from\_acc, to\_acc, amount, label, transactions\_done, bar, anim\_label,  
                      canvas=None, nodes=None, eff\_data=None, ax=None, canvas\_fig=None):  
    try:  
        amount = float(amount)  
        if amount <= 0:  
            messagebox.showerror("Error", "Amount must be greater than zero!")  
            return  
    except ValueError:  
        messagebox.showerror("Error", "Invalid amount!")  
        return  
  
    label.config(text=f"Attempting ${amount} transfer from {[from\_acc.name](http://from_acc.name/)} to {[to\_acc.name](http://to_acc.name/)}...")  
    bar.start(10)  
    threading.Thread(target=animate\_transfer, args=(anim\_label,)).start()  
  
    start\_time = time.time()  
  
    if from\_acc.priority == to\_acc.priority:  
        time.sleep(2)  
        label.config(text=f"Deadlock! Same priority: {[from\_acc.name](http://from_acc.name/)} & {[to\_acc.name](http://to_acc.name/)}", foreground='red')  
        anim\_label.config(image=error\_icon)  
        bar.stop()  
        transactions\_done.append(False)  
        check\_final\_status(transactions\_done)  
        return  
  
    first, second = (from\_acc, to\_acc) if from\_acc.priority < to\_acc.priority else (to\_acc, from\_acc)  
  
    with first.lock:  
        label.config(text=f"{[first.name](http://first.name/)} locked. Waiting for {[second.name](http://second.name/)}...")  
        time.sleep(1)  
        if not second.lock.acquire(timeout=2):  
            label.config(text=f"Deadlock detected! {[from\_acc.name](http://from_acc.name/)} → {[to\_acc.name](http://to_acc.name/)} failed.", foreground='red')  
            bar.stop()  
            anim\_label.config(image=error\_icon)  
            transactions\_done.append(False)  
            check\_final\_status(transactions\_done)  
            return  
        try:  
            from\_acc.withdraw(amount)  
            to\_acc.deposit(amount)  
        finally:  
            second.lock.release()  
  
    end\_time = time.time()  
    duration = round(end\_time - start\_time, 2)  
  
    label.config(text=f" ${amount} transferred {[from\_acc.name](http://from_acc.name/)} → {[to\_acc.name](http://to_acc.name/)}", foreground='green')  
    bar.stop()  
    anim\_label.config(image=success\_icon)  
    update\_balance\_labels()  
    transactions\_done.append(True)  
  
    if canvas and nodes:  
        draw\_arrow(canvas, nodes, from\_acc, to\_acc, amount)  
  
    if eff\_data is not None and ax is not None and canvas\_fig is not None:  
        eff\_data.append(duration)  
        ax.clear()  
        ax.set\_facecolor("#f9f9f9")  
        ax.set\_title("Transaction Time Efficiency", fontsize=14, color='navy', pad=20)  
        ax.set\_xlabel("Transaction #", fontsize=12)  
        ax.set\_ylabel("Time (s)", fontsize=12)  
        ax.tick\_params(colors='black', labelsize=10)  
  
        bars = ax.bar(range(1, len(eff\_data) + 1), eff\_data,  
                      color=plt.cm.viridis([i / len(eff\_data) for i in range(len(eff\_data))]),  
                      edgecolor="black")  
  
        for bar, val in zip(bars, eff\_data):  
            ax.text(bar.get\_x() + bar.get\_width() / 2, bar.get\_height() + 0.05, f"{val:.2f}s",  
                    ha='center', va='bottom', fontsize=10, fontweight='bold')  
  
        ax.grid(axis='y', linestyle='--', alpha=0.5)  
        canvas\_fig.draw()  
  
    check\_final\_status(transactions\_done)  
  
def check\_final\_status(transactions\_done):  
    if len(transactions\_done) == 4:  
        if all(transactions\_done):  
            messagebox.showinfo("Result", " All transactions completed successfully! No Deadlocks.")  
        else:  
            messagebox.showerror("Deadlock Result", f" {transactions\_done.count(False)} transaction(s) failed due to deadlock!")  
  
# -------------------- UI Setup Functions ---------------------  
def update\_balance\_labels():  
    for i, acc in enumerate(bank\_accounts):  
        balance\_labels[i].config(text=f"{[acc.name](http://acc.name/)} [Priority {acc.priority}] Balance: ${acc.balance}")  
  
def build\_transaction\_ui(frame, account\_pairs, transactions\_done\_list,  
                         canvas=None, nodes=None, eff\_data=None, ax=None, canvas\_fig=None):  
    global balance\_labels  
    balance\_labels = []  
    for acc in bank\_accounts:  
        l = ttk.Label(frame, text=f"{[acc.name](http://acc.name/)} [Priority {acc.priority}] Balance: ${acc.balance}", font=("Arial", 14))  
        l.pack(pady=2)  
        balance\_labels.append(l)  
  
    status\_label = ttk.Label(frame, text="", font=("Arial", 13, "bold"))  
    status\_label.pack(pady=8)  
  
    for from\_acc, to\_acc in account\_pairs:  
        trans\_frame = ttk.Frame(frame)  
        trans\_frame.pack(pady=4)  
  
        entry = ttk.Entry(trans\_frame, width=10, font=("Arial", 12))  
        entry.pack(side='left', padx=5)  
  
        bar = ttk.Progressbar(trans\_frame, orient='horizontal', length=100, mode='indeterminate')  
        bar.pack(side='left', padx=5)  
  
        anim = ttk.Label(trans\_frame, image=default\_icon)  
        anim.pack(side='left', padx=5)  
  
        btn = ttk.Button(trans\_frame, text=f"{[from\_acc.name](http://from_acc.name/)} → {[to\_acc.name](http://to_acc.name/)}",  
                         command=lambda f=from\_acc, t=to\_acc, e=entry, p=bar, a=anim:  
                         threading.Thread(target=transfer\_priority, args=(  
                             f, t, e.get(), status\_label, transactions\_done\_list, p, a,  
                             canvas, nodes, eff\_data, ax, canvas\_fig)).start())  
        btn.pack(side='left', padx=5)  
  
# -------------------- Root Setup ---------------------  
root = tk.Tk()  
root.title("Account Transaction Visualizer")  
root.geometry("1000x900")  
root.configure(bg="#e6f2ff")  
  
transfer\_icon1 = ImageTk.PhotoImage(Image.open("transfer1.png").resize((50, 50)))  
transfer\_icon2 = ImageTk.PhotoImage(Image.open("transfer2.png").resize((50, 50)))  
transfer\_icon3 = ImageTk.PhotoImage(Image.open("transfer3.png").resize((50, 50)))  
success\_icon = ImageTk.PhotoImage(Image.open("success.png").resize((50, 50)))  
error\_icon = ImageTk.PhotoImage(Image.open("error.png").resize((50, 50)))  
default\_icon = ImageTk.PhotoImage(Image.open("bank.png").resize((50, 50)))  
  
ttk.Label(root, text=" Account Transaction & Deadlock Visual Simulator", font=("Arial", 20, "bold"), foreground="darkblue").pack(pady=10)  
  
notebook = ttk.Notebook(root)  
notebook.pack(expand=1, fill="both", padx=10, pady=10)  
  
# -------------------- Deadlock Tab ---------------------  
deadlock\_tab = ttk.Frame(notebook)  
notebook.add(deadlock\_tab, text=" Deadlock Demo")  
  
canvas\_deadlock = tk.Canvas(deadlock\_tab, width=600, height=200, bg="white")  
canvas\_deadlock.pack(pady=10)  
  
bank\_accounts = [  
    BankAccount("Account A", 5000, 1),  
    BankAccount("Account B", 3000, 2),  
    BankAccount("Account C", 4000, 2),  
    BankAccount("Account D", 6000, 1)  
]  
node\_coords\_deadlock = {  
    "Account A": (100, 100),  
    "Account B": (250, 50),  
    "Account C": (400, 100),  
    "Account D": (250, 150)  
}  
for acc in bank\_accounts:  
    x, y = node\_coords\_deadlock[[acc.name](http://acc.name/)]  
    canvas\_deadlock.create\_oval(x - 25, y - 25, x + 25, y + 25, fill="lightblue")  
    canvas\_deadlock.create\_text(x, y, text=[acc.name](http://acc.name/))  
  
transactions\_deadlock = [  
    (bank\_accounts[0], bank\_accounts[1]),  
    (bank\_accounts[1], bank\_accounts[2]),  
    (bank\_accounts[2], bank\_accounts[3]),  
    (bank\_accounts[3], bank\_accounts[0])  
]  
transactions\_done\_deadlock = []  
  
fig\_dl, ax\_dl = plt.subplots(figsize=(6, 3))  
canvas\_fig\_dl = FigureCanvasTkAgg(fig\_dl, master=root)  
  
build\_transaction\_ui(deadlock\_tab, transactions\_deadlock, transactions\_done\_deadlock,  
                     canvas\_deadlock, node\_coords\_deadlock,  
                     eff\_data\_deadlock, ax\_dl, canvas\_fig\_dl)  
  
# -------------------- No Deadlock Tab ---------------------  
no\_deadlock\_tab = ttk.Frame(notebook)  
notebook.add(no\_deadlock\_tab, text=" No Deadlock Demo")  
  
canvas\_nodl = tk.Canvas(no\_deadlock\_tab, width=600, height=200, bg="white")  
canvas\_nodl.pack(pady=10)  
  
bank\_accounts = [  
    BankAccount("Account A", 5000, 1),  
    BankAccount("Account B", 3000, 2),  
    BankAccount("Account C", 4000, 3),  
    BankAccount("Account D", 6000, 4)  
]  
node\_coords\_nodl = {  
    "Account A": (100, 100),  
    "Account B": (250, 50),  
    "Account C": (400, 100),  
    "Account D": (250, 150)  
}  
for acc in bank\_accounts:  
    x, y = node\_coords\_nodl[[acc.name](http://acc.name/)]  
    canvas\_nodl.create\_oval(x - 25, y - 25, x + 25, y + 25, fill="lightgreen")  
    canvas\_nodl.create\_text(x, y, text=[acc.name](http://acc.name/))  
  
transactions\_nodl = [  
    (bank\_accounts[0], bank\_accounts[1]),  
    (bank\_accounts[1], bank\_accounts[2]),  
    (bank\_accounts[2], bank\_accounts[3]),  
    (bank\_accounts[3], bank\_accounts[0])  
]  
transactions\_done\_nodl = []  
  
fig\_nodl, ax\_nodl = plt.subplots(figsize=(6, 3))  
canvas\_fig\_nodl = FigureCanvasTkAgg(fig\_nodl, master=root)  
  
build\_transaction\_ui(no\_deadlock\_tab, transactions\_nodl, transactions\_done\_nodl,  
                     canvas\_nodl, node\_coords\_nodl,  
                     eff\_data\_nodl, ax\_nodl, canvas\_fig\_nodl)  
  
# -------------------- Priority Tab ---------------------  
priority\_tab = ttk.Frame(notebook)  
notebook.add(priority\_tab, text=" Priority Info")  
  
for acc in bank\_accounts:  
    ttk.Label(priority\_tab, text=f"{[acc.name](http://acc.name/)} → Priority: {acc.priority}", font=("Arial", 14)).pack(pady=10)  
  
# -------------------- Efficiency Graph Tab ---------------------  
eff\_tab = ttk.Frame(notebook)  
notebook.add(eff\_tab, text=" Efficiency Graph")  
  
ttk.Label(eff\_tab, text=" Deadlock Transactions", font=("Arial", 13, "bold")).pack(pady=(10, 0))  
canvas\_fig\_dl.get\_tk\_widget().pack(in\_=eff\_tab, padx=10, pady=5)  
  
ttk.Label(eff\_tab, text=" No Deadlock Transactions", font=("Arial", 13, "bold")).pack(pady=(10, 0))  
canvas\_fig\_nodl.get\_tk\_widget().pack(in\_=eff\_tab, padx=10, pady=5)  
  
root.mainloop()

|  |  |
| --- | --- |
| https://lh3.google.com/a-/ALV-UjVLW2ffFyLNnFxQEgOW3Tbik4OLoUvdZvvCXEpudSJQjjLdLBVX=s40-p | ReplyForward |