#import gi

import gi

import webbrowser

import os

os.environ["LIBGL\_ALWAYS\_SOFTWARE"] = "1"

os.environ["GDK\_RENDERING"] = "cairo"  # or "gl"

gi.require\_version("Gtk", "4.0")

from gi.repository import Gtk, Gio, Gdk  # Import Gdk for applying the CSS

from PIL import Image, ImageDraw

class MyWindow(Gtk.Window):

    def \_\_init\_\_(self, app):

        super().\_\_init\_\_(title="C-UAS Interface 2025")

        self.set\_default\_size(1000, 600)

        self.set\_application(app)  # Link the window to the application

        def create\_legend\_item(icon\_path, text):

            # Create a horizontal box for the icon and label

            item\_box = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)

            # Create the icon image

            icon = Gtk.Image.new\_from\_file(icon\_path)  # Replace with path to your icon image

            icon.set\_size\_request(30, 30)  # Set the icon size

            # Create the label

            label = Gtk.Label(label=text, halign=Gtk.Align.START)

            # Add the icon and label to the horizontal box

            item\_box.append(icon)

            item\_box.append(label)

            return item\_box

        # Create the main container (a horizontal box) for the window content

        main\_box = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)

        main\_box.set\_hexpand(True)  # Allow horizontal expansion

        main\_box.set\_vexpand(True)  # Allow vertical expansion

        self.set\_child(main\_box)  # Set main\_box as the main widget of the window

        # ------------------------ Left Side Panel -----------------------

        left\_panel = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=0)

        left\_panel.set\_size\_request(300, -1)  # Set a fixed width for the side panel

        left\_panel.add\_css\_class("side-panel")  # Add a CSS class for styling

        left\_panel.set\_vexpand(True)  # Allow the side panel to expand vertically

        # Create a frame (rectangle) for the title and make it expand horizontally

        legend\_frame = Gtk.Frame()

        legend\_frame.add\_css\_class("title-frame")  # Add CSS class for further styling

        legend\_label = Gtk.Label(label="Legend", halign=Gtk.Align.CENTER)

        legend\_frame.set\_child(legend\_label)  # Add the label inside the frame

        left\_panel.append(legend\_frame)

        legend\_box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=5)

        legend\_box.set\_margin\_top(10)

        #Add Items to the Legend

        legend\_box.append(create\_legend\_item(r"C:\Users\C25Jimmy.Nguyen\OneDrive - afacademy.af.edu\Desktop\GUI CAPSTONE\drone\_reboot.png", "Booting Drone"))

        legend\_box.append(create\_legend\_item(r"C:\Users\C25Jimmy.Nguyen\OneDrive - afacademy.af.edu\Desktop\GUI CAPSTONE\DiscoveryDrone.jpg","Discovery Drone"))

        legend\_box.append(create\_legend\_item(r"C:\Users\C25Jimmy.Nguyen\OneDrive - afacademy.af.edu\Desktop\GUI CAPSTONE\RogueDrone.jpg","Rogue Drone"))

        legend\_box.append(create\_legend\_item(r"C:\Users\C25Jimmy.Nguyen\OneDrive - afacademy.af.edu\Desktop\GUI CAPSTONE\FortemRadar.png","Radar"))

        legend\_box.append(create\_legend\_item(r"C:\Users\C25Jimmy.Nguyen\OneDrive - afacademy.af.edu\Desktop\GUI CAPSTONE\GCS.png","Ground Station"))

        left\_panel.append(legend\_box)

        # #Progress Bar for DD

        # progress\_bar\_label = Gtk.Label(label="Discovery Drone Battery (%)", halign=Gtk.Align.CENTER)

        # left\_panel.append(progress\_bar\_label)

        # self.progress\_bar = Gtk.ProgressBar()

        # self.progress\_bar.set\_margin\_top(10)

        # self.progress\_bar.set\_margin\_bottom(10)

        # left\_panel.append(self.progress\_bar)

        # #Progress Bar for RD

        # progress\_bar\_label = Gtk.Label(label="Rogue Drone Battery (%)", halign=Gtk.Align.CENTER)

        # left\_panel.append(progress\_bar\_label)

        # self.progress\_bar2 = Gtk.ProgressBar()

        # self.progress\_bar2.set\_margin\_top(10)

        # self.progress\_bar2.set\_margin\_bottom(10)

        # left\_panel.append(self.progress\_bar2)

        # Create a frame (rectangle) for the title and make it expand horizontally

        status\_frame = Gtk.Frame()

        status\_frame.add\_css\_class("title-frame")  # Add CSS class for further styling

        status\_label = Gtk.Label(label="Status Panel", halign=Gtk.Align.CENTER)

        status\_frame.set\_child(status\_label)  # Add the label inside the frame

        left\_panel.append(status\_frame)

        status\_box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=5)

        status\_box.set\_margin\_top(10)

        # Add the side panel to the main container

        main\_box.append(left\_panel)

        # ----------------------- Main Content Area (Center) ----------------

        content\_area = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=0)

        content\_area.add\_css\_class("content-area")  # Add a CSS class for styling

        content\_area.set\_hexpand(True)  # Allow the content area to expand horizontally

        content\_area.set\_vexpand(True)  # Allow the content area to expand vertically

        # Button to open Google Maps in the system browser

        open\_maps\_button = Gtk.Button(label="Open Google Maps Satellite View")

        open\_maps\_button.connect("clicked", self.open\_maps)

        content\_area.append(open\_maps\_button)

        #---------------------GUI LatLong Grids-----------------

        #Load static image map

        image\_path = r"C:\Users\C25Jimmy.Nguyen\OneDrive - afacademy.af.edu\Desktop\GUI CAPSTONE\Test2Map.png"

        map\_image = Image.open(image\_path)

        draw = ImageDraw.Draw(map\_image)

        image\_width, image\_height = map\_image.size

        # Lat/Long for the four corners of the image

        # lat1, lon1 = 39.01897, -104.89435  # Top-Left

        # lat2, lon2 = 39.01897, -104.89431  # Top-Right

        # lat3, lon3 = 39.01739, -104.89435  # Bottom-Left

        # lat4, lon4 = 39.01739, -104.89431  # Bottom-Right

        #New test coordinates

        lat1, lon1 = 39.019045, -104.894301  # Top-Left

        lat2, lon2 = 39.019045, -104.892113  # Top-Right

        lat3, lon3 = 39.017430, -104.894301  # Bottom-Left

        lat4, lon4 = 39.017430, -104.892113  # Bottom-Right

        ## Correctly calculated lat\_per\_pixel and lon\_per\_pixel

        lat\_range = lat1 - lat3  # Should be positive

        lon\_range = lon2 - lon1  # Should be positive

        lat\_per\_pixel = lat\_range / image\_height

        lon\_per\_pixel = lon\_range / image\_width

        # Check for zero range and raise an error if detected

        if lat\_range == 0 or lon\_range == 0:

            raise ValueError("Latitude or Longitude range is zero. Please adjust coordinates.")

        # Function to convert lat/long to pixel coordinates

        def latlon\_to\_pixel(latitude, longitude):

             # Check if the coordinate is within bounds

            if not (lat3 <= latitude <= lat1) or not (lon1 <= longitude <= lon2):

                print(f"Warning: Latitude {latitude} or Longitude {longitude} is out of the image bounds.")

                return None

            # Calculate pixel position

            pixel\_x = (longitude - lon1) / lon\_per\_pixel

            pixel\_y = (lat1 - latitude) / lat\_per\_pixel

            print(f"Lat/Lon: ({latitude}, {longitude}) -> Pixel: ({int(pixel\_x)}, {int(pixel\_y)})")

            return int(pixel\_x), int(pixel\_y)

        # Check that coordinates fall within adjusted bounds

        #latitude, longitude = 39.01818, -104.89433  # CENTER COORD

        #latitude, longitude = 39.01897, -104.89433 # TOP CENTER COORD

        #latitude, longitude = 39.01739, -104.89433 # BOTTOM CENTER COORD

        #latitude, longitude = 39.01818, -104.89435 # LEFT CENTER COORD

        #latitude, longitude = 39.01818, -104.89431 # RIGHT CENTER COORD

        #Radar Coord Data

        latitude, longitude = 39.01839, -104.89353

        if lat3 <= latitude <= lat1 and lon1 <= longitude <= lon2:

            x, y = latlon\_to\_pixel(latitude, longitude)

            draw.ellipse((x - 5, y - 5, x + 5, y + 5), fill="red", outline="black")

        else:

            print("Adjusted coordinates are still out of bounds.")

        # Save or display the result

        map\_image.save(r"C:\Users\C25Jimmy.Nguyen\OneDrive - afacademy.af.edu\Desktop\GUI CAPSTONE\NewTest2Map.png")

        map\_image.show()

        #-------------------------------------------------------

        # Load the image

        #image = Gtk.Image.new\_from\_file("C:\\Users\\C25Jimmy.Nguyen\\OneDrive - afacademy.af.edu\\Desktop\\GUI CAPSTONE\\Screenshot 2024-10-04 224853.png")

        image = Gtk.Picture.new\_for\_filename(r"C:\Users\C25Jimmy.Nguyen\OneDrive - afacademy.af.edu\Desktop\GUI CAPSTONE\Test2Map.png")

        # Ensure the image expands and fills the available space

        image.set\_hexpand(True)

        image.set\_vexpand(True)

        image.set\_content\_fit(Gtk.ContentFit.FILL)  # Make the image fill the container

        # Add the image to the content area

        content\_area.append(image)

        # Add the content area to the main container

        main\_box.append(content\_area)

        # ----------------------- Right Side Panel ----------------------------

        right\_panel = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=0)

        right\_panel.set\_size\_request(300, -1)  # Set a fixed width for the side panel

        right\_panel.add\_css\_class("side-panel")  # Add a CSS class for styling

        right\_panel.set\_vexpand(True)  # Allow the side panel to expand vertically

        # Create a frame (rectangle) for the title and make it expand horizontally

        title\_frame = Gtk.Frame()

        # title\_frame.set\_hexpand(True)  # Make the frame expand horizontally

        title\_frame.add\_css\_class("title-frame")  # Add CSS class for further styling

        # Create a label for the title and add it inside the frame

        right\_panel\_title = Gtk.Label(label="Controls", halign=Gtk.Align.CENTER)

        title\_frame.set\_child(right\_panel\_title)  # Add the label inside the frame

        right\_panel.append(title\_frame)

        # Create a button and add it to the right panel

        button\_in\_right\_panel = Gtk.Button(label="Toggle Drone Path")

        button\_in\_right\_panel.set\_margin\_top(10)

        right\_panel.append(button\_in\_right\_panel)

        button2\_in\_right\_panel = Gtk.Button(label="Log Data")

        button2\_in\_right\_panel.set\_margin\_top(10)

        right\_panel.append(button2\_in\_right\_panel)

        button3\_in\_right\_panel = Gtk.Button(label="Input Lat/Long")

        button3\_in\_right\_panel.set\_margin\_top(10)

        right\_panel.append(button3\_in\_right\_panel)

        button4\_in\_right\_panel = Gtk.Button(label="Reload Map")

        button4\_in\_right\_panel.set\_margin\_top(10)

        right\_panel.append(button4\_in\_right\_panel)

        button5\_in\_right\_panel = Gtk.Button(label="Freeze")

        button5\_in\_right\_panel.set\_margin\_top(10)

        right\_panel.append(button5\_in\_right\_panel)

        button6\_in\_right\_panel = Gtk.Button(label="Camera Stream")

        button6\_in\_right\_panel.set\_margin\_top(10)

        button7\_in\_right\_panel = Gtk.Button(label="Follow-me Mode")

        button7\_in\_right\_panel.set\_margin\_top(10)

        right\_panel.append(button7\_in\_right\_panel)

        main\_box.append(right\_panel)

        # Apply the custom CSS styles

        self.apply\_css()

    def apply\_css(self):

        # Load CSS provider

        css\_provider = Gtk.CssProvider()

        # Define CSS styles for the side panel and content area

        css = b"""

        .side-panel {

            border: 2px solid black;

            padding: 10px;

        }

        .content-area {

            border: 2px solid blue;

            padding: 10px;

        }

        .title-frame {

            background-color: #a9a9a9;  /\* Set background color \*/

            border: 2px solid black;    /\* Add border to the frame \*/

            padding: 5px;

        }

        label {

            font-weight: bold;  /\* Make the label text bold \*/

        }

        """  # Make sure the triple-quoted string is properly closed here

        # Load the CSS

        css\_provider.load\_from\_data(css)

        # Get the default display from Gdk (not Gtk)

        display = Gdk.Display.get\_default()

        # Apply the CSS provider to the display's default screen

        Gtk.StyleContext.add\_provider\_for\_display(display, css\_provider, Gtk.STYLE\_PROVIDER\_PRIORITY\_APPLICATION)

    def open\_maps(self, widget):

        # URL to open Google Maps in satellite view

        google\_maps\_url = "https://www.google.com/maps/place/Stillman+Field/@39.0090343,-104.8828153,255m/data=!3m1!1e3!4m6!3m5!1s0x8713537f7d0f542b:0x9f68218ee71a4b94!8m2!3d39.0088982!4d-104.8818144!16s%2Fg%2F11stld1zr5?entry=ttu&g\_ep=EgoyMDI0MTAwMi4xIKXMDSoASAFQAw%3D%3D"

        webbrowser.open(google\_maps\_url)

class MyApp(Gtk.Application):

    def \_\_init\_\_(self):

        super().\_\_init\_\_(application\_id="org.example.myapp", flags=Gio.ApplicationFlags.FLAGS\_NONE)

    def do\_activate(self):

        # Create and present the window when the application is activated

        win = MyWindow(self)

        win.present()

def main():

    app = MyApp()

    app.run(None)

if \_\_name\_\_ == "\_\_main\_\_":

    main()