🎖️ The "By Jove" WITE2 Planning Manual

Goal: Create a 1:1 pixel-perfect tactical overlay with automated CP counting.

Phase 0: Required Software (The Engine)

Before you begin, you need these three free tools installed on your PC:

1. **Python:** Download from python.org.
    - *IMPORTANT:* During installation, check the box that says **"Add Python to PATH"**.
2. **Inkscape:** Download from inkscape.org. This is your tactical planning table.
3. **Required Libraries:** You need three specific "parts" for Python to handle the game data.
    - Open your Windows search bar, type `cmd`, and press Enter.
    - Paste the following command and press Enter: `pip install pandas numpy scipy`

Phase 1: The Tactical Map (Source)

1. **Go to the Drive Folder:** Map Source Location
2. **Download the Image:** Right-click the map file and select **Download**.
    - **CRITICAL:** Do *not* copy-paste. You must download the file to preserve the **22,760 x 21,040** pixel ratio.
3. **Open in Inkscape:** Launch Inkscape and use **File > Open** to select the downloaded map image.
4. **Lock the Base:** Open the Layers panel (**Ctrl+Shift+L**), rename the layer to "Game Map," and click the **Lock** icon.

Phase 2: Setting up the Macro Folder

1. **Create Folder:** Create a folder on your **Desktop** named `Planning Map`.
2. **Create the Script with Notepad:**
    - Copy the code block below.
    - Open **Notepad** and paste the code.
    - Go to **File > Save As...**
    - Change "Save as type" to **All Files (.)**.
    - Name the file `make_overlay.py` and save it into your `Planning Map` folder.
3. **Add Your Data:** Export your turn's **Unit-Main** and **Leader-Leaders** CSVs from the game and drop them into this same folder.

Python

```
import os
import pandas as pd
```

```
import numpy as np
from scipy.spatial import ConvexHull

# --- THE GEM PARAMETERS (STRICT ALIGNMENT) ---
MAP_W, MAP_H = 22760, 21040
COLS, ROWS = 378, 352
DX, DY = MAP_W / COLS, MAP_H / ROWS

def get_corps_color(name):
    name_upper = str(name).upper()
    if any(keyword in name_upper for keyword in ["MOT.", "PANZER", "PZ."]):
        return "#4682B4" # Slate Blue (Mobile)
    return "#2E8B57"     # Sea Green (Infantry)

def get_pixel_pos(x, y):
    px = (x * DX) + (DX / 2)
    py = (y * DY) + (DY / 2)
    if int(x) % 2 != 0: py += (DY / 2)
    return px, py

def create_final_overlay():
    u_file = next((f for f in os.listdir('.') if 'Unit-Main' in f and
f.endswith('.csv')), None)
    l_file = next((f for f in os.listdir('.') if 'Leader-Leaders' in f and
f.endswith('.csv')), None)
    if not u_file:
        print("ERROR: Unit-Main CSV not found in this folder."); return

    try:
        df = pd.read_csv(u_file, low_memory=False)
        admin_map = pd.read_csv(l_file).set_index('Unit')['Adm'].to_dict()
if l_file else {}

        # FILTER: Only Combat units (excludes Railroad/Air/Labor clutter)
        ignored_types = ['AirBase', 'Const', 'Labor', 'Ferry', 'Fort']
        df_map = df[(df['ThBox'] == 'MAP') &
(~df['Type'].isin(ignored_types))].copy()

        df_map['X'] = pd.to_numeric(df_map['X'], errors='coerce')
        df_map['Y'] = pd.to_numeric(df_map['Y'], errors='coerce')
        df_map = df_map.dropna(subset=['X', 'Y'])

        corps_hqs = df_map[df_map['Size'] == 'XXX']
        corps_names = set(corps_hqs['Unit Name'].unique())
        corps_groups = {name: [get_pixel_pos(row['X'], row['Y'])] for name,
row in corps_hqs.set_index('Unit Name').iterrows()}

        usage = {name: 0 for name in corps_names}
        for _, row in df_map.iterrows():
            parent = str(row['HHQ']).strip()
            if parent in corps_names:
                sz = str(row['Size'])
                if sz in ['XX', 'X']: usage[parent] += 2
                elif sz == 'III': usage[parent] += 1

        elements = []

        # 1. Tactical Clouds
        for name, points in corps_groups.items():
            if len(points) < 2: continue
            pts = np.unique(np.array(points), axis=0)
```

```
            try:
                if len(pts) >= 3:
                    hull = ConvexHull(pts)
                    d = "M " + " L ".join([f"{p[0]:.2f},{p[1]:.2f}" for p
in pts[hull.vertices]]) + " Z"
                else:
                    d = f"M {pts[0][0]},{pts[0][1]} L {pts[-1][0]},{pts[-
1][1]}"
                elements.append(f'<path d="{d}"
fill="{get_corps_color(name)}" opacity="0.25"
stroke="{get_corps_color(name)}" stroke-width="140" stroke-linejoin="round"
/>')
            except: pass

        # 2. Units & HQs
        for _, row in df_map.iterrows():
            sz = str(row['Size'])
            if sz not in ['XX', 'X', 'III']: continue
            px, py = get_pixel_pos(row['X'], row['Y'])
            parent = str(row['HHQ']).strip()

            if parent in corps_names:
                corps_groups[parent].append((px, py))
                elements.append(f'<circle cx="{px:.2f}" cy="{py:.2f}" r="7"
fill="{get_corps_color(parent)}" />')
            else:
                elements.append(f'<circle cx="{px:.2f}" cy="{py:.2f}"
r="25" fill="#FF0000" opacity="0.4" />')
                elements.append(f'<circle cx="{px:.2f}" cy="{py:.2f}" r="8"
fill="#FF0000" stroke="white" stroke-width="2" />')

        # 3. Permanent Labels
        for _, row in corps_hqs.iterrows():
            name_raw = str(row['Unit Name'])
            px, py = get_pixel_pos(row['X'], row['Y'])
            clr = get_corps_color(name_raw)
            adm = admin_map.get(name_raw, 5)
            lim = 10 if (adm >= 7 or any(x in name_raw for x in ["Mot",
"Panzer", "Pz"])) else 9
            t_clr = "#FF0000" if usage[name_raw] > lim else clr

            elements.append(f'<circle cx="{px:.2f}" cy="{py:.2f}" r="18"
stroke="{clr}" stroke-width="6" fill="white" />')
            elements.append(f'<g transform="translate({px:.2f},
{py:.2f})">'
                            f'<text y="35" font-family="Arial" font-
size="21" text-anchor="middle" fill="{clr}" font-weight="bold"
stroke="white" stroke-width="3" paint-order="stroke">{name_raw.replace("&",
"&amp;")}</text>'
                            f'<text y="58" font-family="Arial" font-
size="18" text-anchor="middle" fill="{t_clr}" font-weight="bold"
stroke="white" stroke-width="3" paint-
order="stroke">{usage[name_raw]}/{lim}</text></g>')

        header = f'<svg width="{MAP_W}px" height="{MAP_H}px" viewBox="0 0
{MAP_W} {MAP_H}" xmlns="http://www.w3.org/2000/svg">\n'
        with open('Planning_Overlay.svg', "w") as f:
            f.write(header + f'<rect width="{MAP_W}" height="{MAP_H}"
fill="none" pointer-events="none" />\n')
            f.write("\n".join(elements) + '</svg>')
        print("Success! Created Planning_Overlay.svg.")
```

```
    except Exception as e: print(f"ERROR: {e}")

if __name__ == "__main__":
    create_final_overlay()
    input("\nPress Enter to close...")
```

Phase 3: Deployment

1. **Run:** Double-click `make_overlay.py` in your desktop folder. A black window will appear; confirm success and press **Enter**.
2. **The Result:** You will notice a new file has appeared in your folder: **`Planning_Overlay.svg`**. This is the file you will use in Inkscape.
3. **Import:** In Inkscape, go to **File > Import** and select `Planning_Overlay.svg`.
4. **Snap:** In the top toolbar, set the overlay position to **X: 0, Y: 0**.
5. **Result:** A map is created with "Corps" footprints overlaid on the Planning Map. Footprints are green for infantry and blue for motorized. Each Corps shows its current Command Limit and Command Status. Units unattached directly to a Corps are shown in red.

NOTE: Use the scrollbars to move the map around. If you accidentally drag the overlay with your mouse, simply reset the **X, Y** coordinates to **0, 0**.

After each turn: Export the new Commanders Report files and copy Unit-Main and Leader-Leaders and replace the old ones in the folder. Run the script again to generate the updated overlay.