

Simple AAC board interface design

Exported on 12/4/2025 at 11:33:51 EST from Cursor (2.0.77)

User

In Matts Workspace, I want a super simple, minimal code AAC board interface that has buttons that have images from the @arasaac_synset_mapping_20251106_130530.csv , the buttons are generated by passing the board a list of synset IDs

Cursor

Examining the CSV structure and Matt's Workspace files.

Creating a minimal AAC board interface that takes synset IDs and displays buttons with images.
Using Python with tkinter.

Creating a minimal AAC board interface:

```
import tkinter as tk
from tkinter import ttk
import pandas as pd
from PIL import Image, ImageTk
import requests
from io import BytesIO

class AACBoard:
    def __init__(self, csv_path, synset_ids):
        """
        Initialize the AAC board interface.

        Args:
            csv_path: Path to the arasaac_synset_mapping CSV file
            synset_ids: List of synset IDs to display as buttons
        """
        self.synset_ids = synset_ids

        # Load CSV and create mapping
```

```

df = pd.read_csv(csv_path)
# Get first occurrence of each synset (or filter by provided synset_ids)
self.synset_map = {}
for synset in synset_ids:
    matches = df[df['synset'] == synset]
    if not matches.empty:
        row = matches.iloc[0]
        self.synset_map[synset] = {
            'image_url': row['image_url'],
            'keyword': row['primary_keyword'],
            'pictogram_id': row['pictogram_id']
        }

# Create main window
self.root = tk.Tk()
self.root.title("AAC Board")
self.root.geometry("800x600")

# Create main frame
main_frame = ttk.Frame(self.root, padding="10")
main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

# Configure grid weights
self.root.columnconfigure(0, weight=1)
self.root.rowconfigure(0, weight=1)
main_frame.columnconfigure(0, weight=1)
main_frame.rowconfigure(0, weight=1)

# Create canvas with scrollbar
canvas = tk.Canvas(main_frame)
scrollbar = ttk.Scrollbar(main_frame, orient="vertical", command=canvas.yview)
scrollable_frame = ttk.Frame(canvas)

scrollable_frame.bind(
    "<Configure>",
    lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
)

canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
canvas.configure(yscrollcommand=scrollbar.set)

# Load and display buttons
self.buttons = []
self.load_buttons(scrollable_frame)

canvas.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))
scrollbar.grid(row=0, column=1, sticky=(tk.N, tk.S))
main_frame.rowconfigure(0, weight=1)
main_frame.columnconfigure(0, weight=1)

# Bind mousewheel to canvas
def on_mousewheel(event):
    canvas.yview_scroll(int(-1*(event.delta/120)), "units")
    canvas.bind_all("<MouseWheel>", on_mousewheel)

```

```
def load_image_from_url(self, url):
    """Load image from URL and resize for button."""
    try:
        response = requests.get(url, timeout=5)
        img = Image.open(BytesIO(response.content))
        img = img.resize((100, 100), Image.Resampling.LANCZOS)
        return ImageTk.PhotoImage(img)
    except Exception as e:
        print(f"Error loading image from {url}: {e}")
        # Return a placeholder
        img = Image.new('RGB', (100, 100), color='gray')
        return ImageTk.PhotoImage(img)

def load_buttons(self, parent):
    """Load and display buttons for each synset."""
    row = 0
    col = 0
    cols_per_row = 4

    for synset_id in self.synset_ids:
        if synset_id not in self.synset_map:
            print(f"Warning: Synset {synset_id} not found in CSV")
            continue

        info = self.synset_map[synset_id]

        # Create button frame
        btn_frame = ttk.Frame(parent, padding="5")
        btn_frame.grid(row=row, column=col, padx=5, pady=5)

        # Load image
        photo = self.load_image_from_url(info['image_url'])

        # Create button
        btn = tk.Button(
            btn_frame,
            image=photo,
            text=info['keyword'],
            compound=tk.TOP,
            width=120,
            height=130,
            command=lambda s=synset_id: self.on_button_click(s)
        )
        btn.image = photo # Keep a reference
        btn.pack()

        self.buttons.append(btn)

        # Update grid position
        col += 1
        if col >= cols_per_row:
            col = 0
            row += 1
```

```

def on_button_click(self, synset_id):
    """Handle button click."""
    info = self.synset_map.get(synset_id, {})
    keyword = info.get('keyword', synset_id)
    print(f"Clicked: {keyword} ({synset_id})")
    # You can extend this to handle speech synthesis, etc.

def run(self):
    """Start the GUI."""
    self.root.mainloop()

if __name__ == "__main__":
    # Example usage
    csv_path = "../Kaitlin's Workspace/arasaac_synset_mapping_20251106_130530.csv"

    # Example synset IDs
    synset_ids = [
        "04222469-n", # pavement
        "07951744-n", # water
        "04125115-n", # carpet
        "03944520-n", # pillow
        "02703861-n", # ambulance
        "04099721-n", # ring
    ]
    board = AACBoard(csv_path, synset_ids)
    board.run()

```

Checking for lint issues and creating a simple example:

Created a minimal AAC board interface. It includes:

1. **AACBoard class:** Creates a GUI with buttons for synset IDs
2. **Image loading:** Fetches images from the ARASAAC API URLs in the CSV
3. **Grid layout:** Displays buttons in a 4-column grid with scroll
4. **Button click handling:** Logs clicks (you can extend this for speech, etc.)

Features:

- Loads the CSV and maps synset IDs to image URLs and keywords
- Displays buttons with images and keyword labels
- Scrollable interface for many buttons
- Simple to use: pass a list of synset IDs

To use it:

```
from aac_board import AACBoard
```

```
csv_path = "../Kaitlin's Workspace/arasaac_synset_mapping_20251106_130530.csv"
synset_ids = ["04222469-n", "07951744-n", "04125115-n"] # Your synset IDs

board = AACBoard(csv_path, synset_ids)
board.run()
```

The script includes example synset IDs. To run it, you'll need:

- pandas
- Pillow
- requests
- tkinter (usually included with Python)

If the ARASAAC API requires a different image URL format, I can adjust it. Test it and let me know if you need changes.

```
from aac_board import AACBoard

csv_path = "../Kaitlin's Workspace/arasaac_synset_mapping_20251106_130530.csv"
synset_ids = ["04222469-n", "07951744-n", "04125115-n"] # Your synset IDs

board = AACBoard(csv_path, synset_ids)
board.run()
```

User

looks good. The button images load correctly and the clicks are registered in the terminal.

Cursor

Good to hear the interface is working. The buttons load images from the ARASAAC API and clicks register in the terminal.

If you want changes or additions, such as:

- Speech synthesis when buttons are clicked
- Customizable board layouts
- Saving/loading board configurations
- Keyboard shortcuts
- Visual feedback on clicks

Let me know and I can implement them.

User

What do you think I should add next?

Cursor

Suggested next steps, in priority order:

1. Speech synthesis (highest priority) – speaking the keyword on button click; using pyttsx3 or gTTS.
2. Sentence building – collect selected buttons, show the phrase, and speak it all at once; add a "Clear" button.
3. Visual feedback – highlight selected buttons and show a sentence area.
4. Board configuration – save/load board layouts for different contexts (e.g., "Breakfast", "Activities").
5. Multiple board support – switch between boards or categories.
6. Accessibility – keyboard navigation and larger button/text size options.

I recommend starting with speech synthesis, then sentence building. Should I implement these next?

No speech synthesis found in the codebase. For an AAC board, speech output is the priority.

Recommended order:

1. **Speech synthesis** – Speak words when buttons are clicked
2. **Sentence building** – Build phrases by clicking multiple buttons, then speak the full sentence
3. **Visual feedback** – Show selected buttons and a sentence preview area

For Python TTS options:

- pyttsx3 – offline, cross-platform, no internet needed
- gTTS (Google Text-to-Speech) – online, better quality, requires internet

I recommend starting with pyttsx3 for offline use. Should I add speech synthesis and sentence building next?

