Aalto-yliopisto
Aalto-universitetet
Aalto University

ELEC-E7852
Computational Interaction and Design

Assignment A3a

Saliency models

Aitor Urruticoechea Puig
aitor.urruticoecheapuig@aalto.fi
Student N°101444219
November 2024

Notice

Acknowledgement

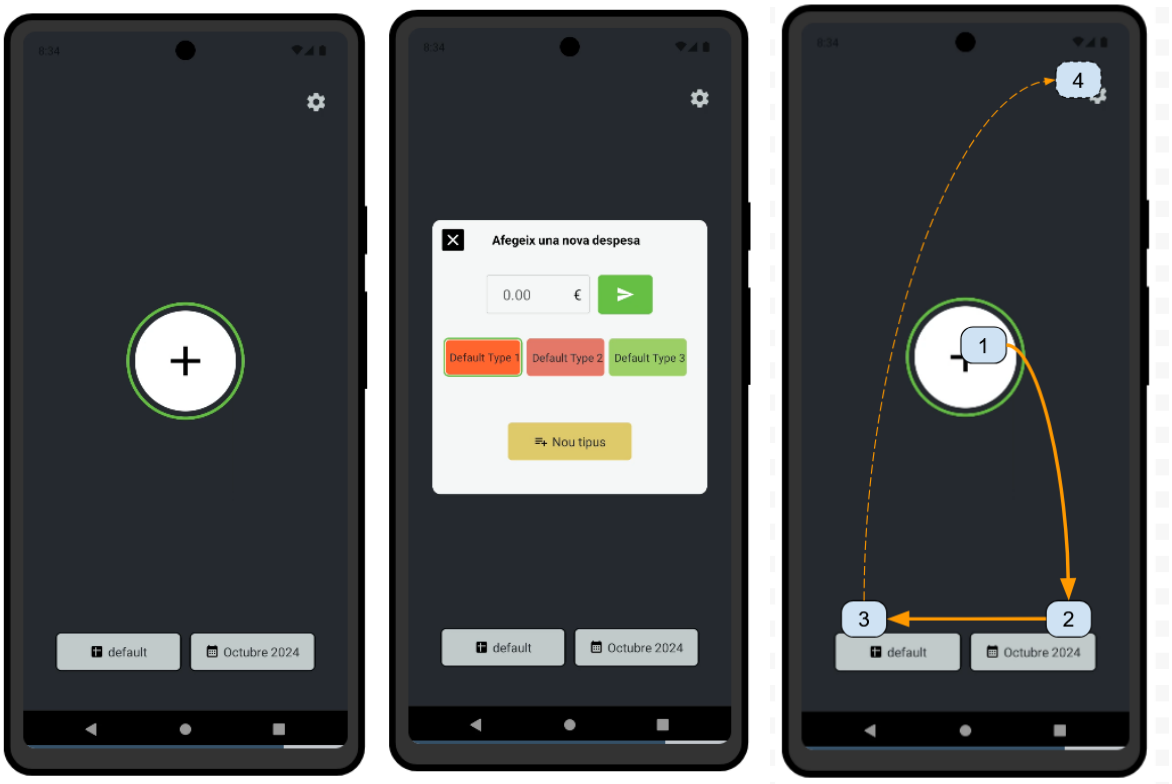
Parts of the code implemented in this work were generated with the assistance of GitHub Copilot. None of its proposals were, however, employed without refinement and necessary tweaks to adapt it to the nuances of the tasks at hand; making it impossible to identify and subsequently mark which lines of code were human or machine-made, for many were the fruit of the combination of both.

Contents

List of Figures

1 Case description

The goal for this assignment is to algorithmically change the display of the chosen case to achieve a visual flow that guides the user through the desired paths in checking three elements. "This requires that your algorithm can 1) generate candidate designs and 2) use EyeFormer to assess if the desired visual flow has been achieved." Again, the personal React-native project DespesApp for spending tracking will be using as a baseline (Figure 1). As a case study, let us take a look to the home screen for the app (Figure 1a). It is quite a simple landing page, but it has more than enough elements for us to be able to play around with different configurations. In the centre, there is the main "add new spending" button that leads to the screen shown in Figure 1b. In the bottom, there are the two supporting buttons: the right one is the second one we want the user to see, as it allows for changing the timestamp for the spending registry (which month/year should the spending action be registered in); while the left one is a minor settings feature to set which spreadsheet or database should the spending action be registered in. Finally, in the top-right corner, there is the settings button, which should be mostly ignored by every user unless they are specifically looking for it. All in all, the desired visual flow should ultimately result in something along the lines of what is shown in Figure 1c.



(a) Home screen (b) "Add new spending" window. (c) Desired visual flow.

Figure 1: "DespesApp" spending tracking app overview.

2 UI representation for optimization and Objective Function

To simplify the optimization parameters; a display creation function has been implemented that helps in iterating through designs. This function lets the user play around with the three main buttons and its respective positions and sizes, while leaving colours and other attention features unchanged. This does numerically simplify the process, as it leaves us with 9 parameters to tune: for each of the three buttons, there is the x and y coordinates, and the scale factor. To make it even easier for the optimizer, every one of these parameters have been phrased so that they are to be expressed as a percentage, making it a direct correlation to $0 - 1$ values traditionally handled better by optimizers.

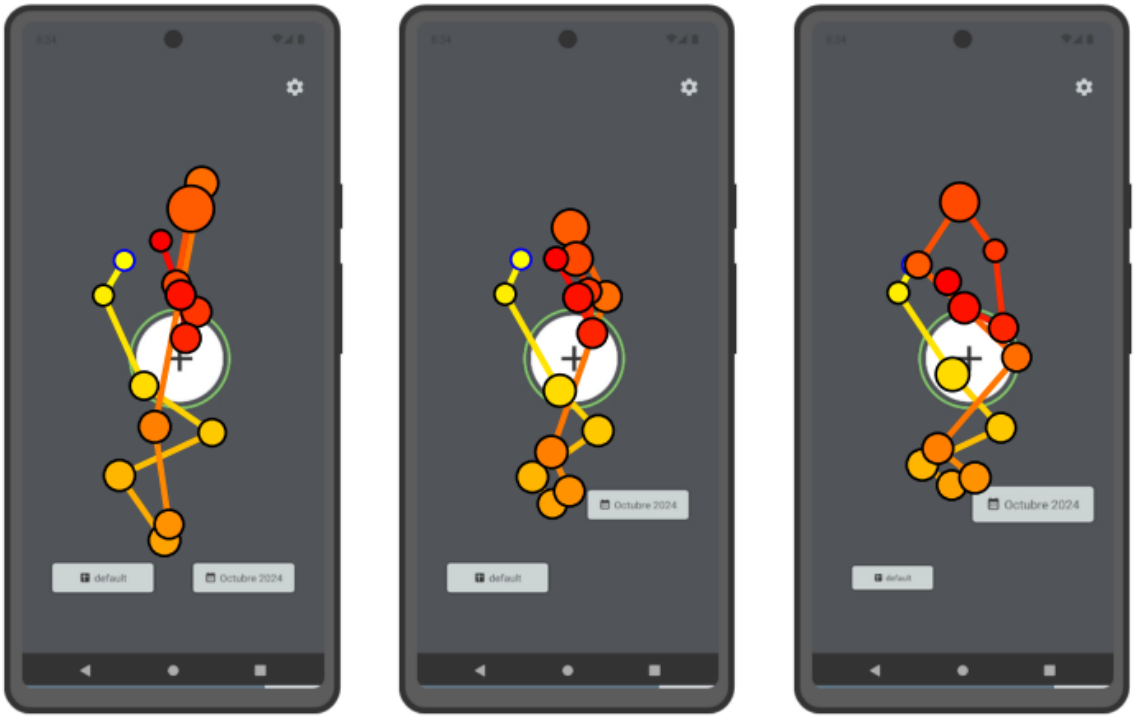
```python
from PIL import Image

def create_display(background_path, sticker_paths, coordinates, sizes):
    # Open the background image
    background = Image.open(background_path).convert("RGBA")
    bg_width, bg_height = background.size

    # Loop through each sticker image
    for i, sticker_path in enumerate(sticker_paths):
        sticker = Image.open(sticker_path).convert("RGBA")
        max_size = sizes[i]

        aspect_ratio = sticker.width / sticker.height
        if sticker.width > sticker.height:
            new_width = max_size
            new_height = int(max_size / aspect_ratio)
        else:
            new_height = max_size
            new_width = int(max_size * aspect_ratio)

        sticker = sticker.resize((new_width, new_height), Image.LANCZOS)

        # so that the corrdinates refer to the centerpoint of each sticker
        x = int(coordinates[i][0] * bg_width / 100) - new_width // 2
        y = int(coordinates[i][1] * bg_height / 100) - new_height // 2

        background.paste(sticker, (x, y), sticker)

    return background
```

This image can, of course, be used as an input for the Visual Saliency model of our choice. Using the one provided in class (available in Jupyter). For that a good objective function, that unfortunately it was not possible to iterate upon for optimization, could look something like the following code, where the approximate desire path expressed in coordinates is used as a quantifiable objective (distance point-to-point), thus resulting in a numerical value that the desired optimizer can try to minimize as much as possible.

```python
def calculate_distance(scanpath, objectives):
    n = min(len(scanpath), len(objectives))
    return np.sqrt(np.sum((scanpath[:n] - objectives[:n])**2, axis=1)).mean()

# Define the objective function for Bayesian Optimization
def objective_function(params, background_path, sticker_paths, scanpath_csv_path, objectives):

    # Extract parameters
    coordinates = [(params[0].item(), params[1].item()), (params[2].item(), params[3].item()), (params[4].item(), params[5].item())]
    sizes = [int(params[6].item()), int(params[7].item()), int(params[8].item())]

    # Generate an image with current parameters and a scanpath CSV

    add_stickers_to_background(background_path, sticker_paths, coordinates, sizes).save("temp_image.png")
    generate_scanpath("temp_image.png", sticker_paths, coordinates, sizes, scanpath_csv_path) # To be adapted

    # Load the scanpath for the generated image and calculate distance to objectives
    scanpath = load_scanpath(scanpath_csv_path, "temp_image.png")

    return calculate_distance(scanpath, objectives)

```

3 Results and Conclusion

While the full optimization was not possible to implement, the potential behind the proposed approach is clear when one looks at some of the manually iterated results, where the display is procedurally changed to obtain different arrangements of buttons and sizes (Figure 2).



(a) Initial layout scanpath results (b) Coordinate-iterated scanpath. (c) Plus size-iterated scanpath.

Figure 2: Manual iteration results.