Window: music visualizer designed for workplace

"Window" is a dynamic digital window that brings the vibrant outside world through audio-visual interpretations.

While music visualizers usually demand a full-screen immersion, this luxury is impractical during focused work hours. The slim proportion and the subtle amplitude of animations are deliberately designed to allow users to maintain peripheral awareness without feeling disrupted.

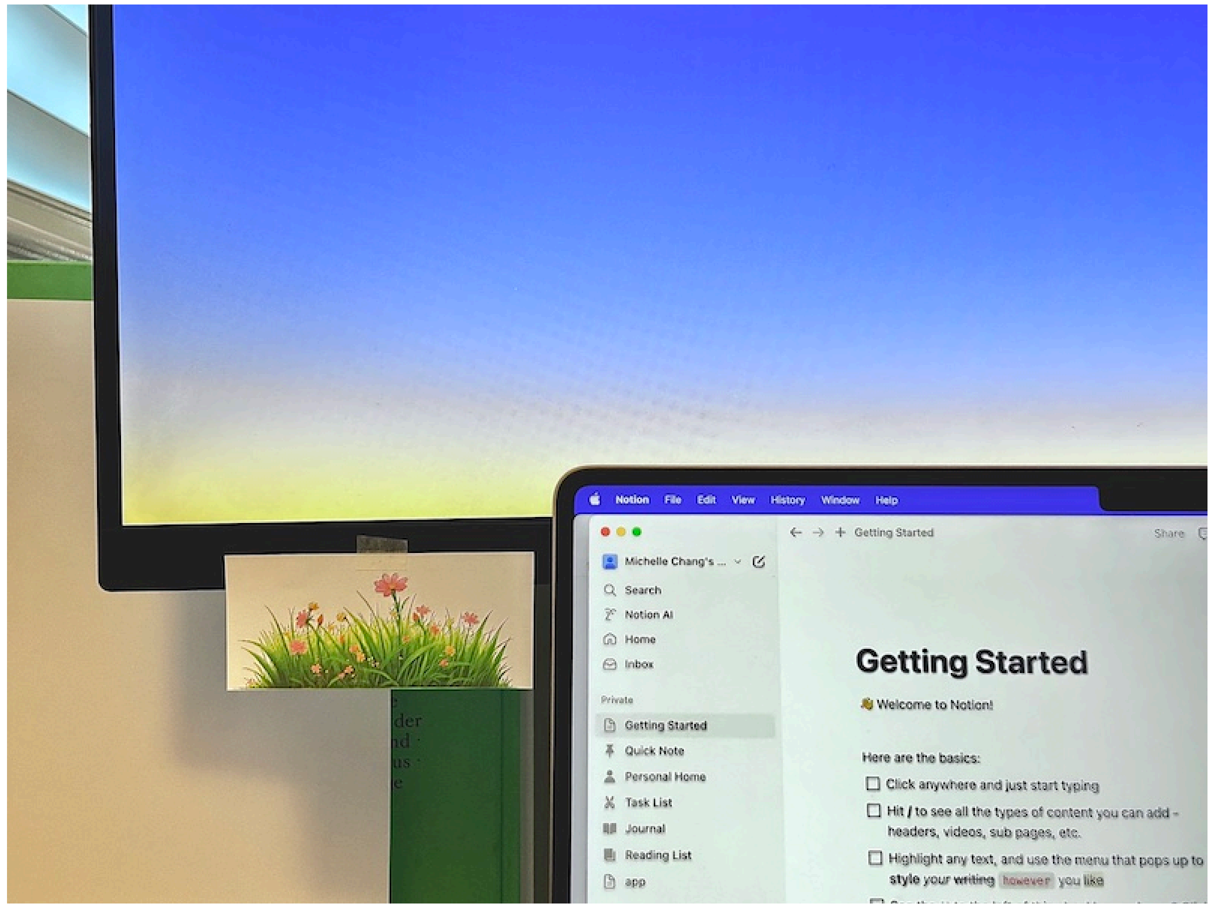
Independent project
June 2024

User research

The Window emerged from observing a common challenge in workplaces: the disconnect between indoor office environments and the natural world outside. To validate this observation, I designed a two-week guerrilla research study exploring how subtle nature elements might impact workplace wellbeing.

I created a low-fidelity approach that allowed for rapid testing, by placing printed copies of nature graphics next to their workspaces.

This minimal intervention revealed valuable insights about how subtle natural elements could enhance the workplace environment.

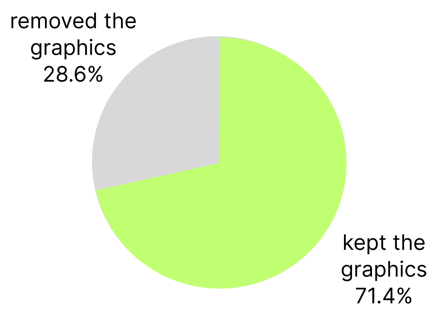


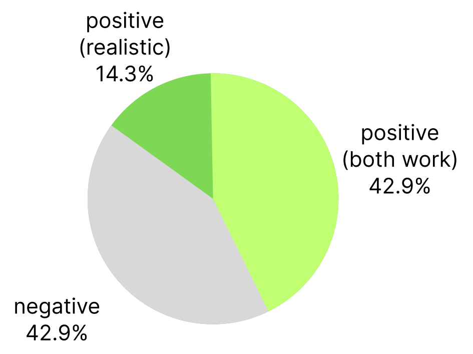
Observations

- Users frequently glanced at the installations in between tasks, indicating potential stress-relief function
- Users unconsciously adjusted the installations to preserve workspace efficiency

Design implications

- Visual elements must maintain a minimal footprint to prevent cognitive load during focused work periods





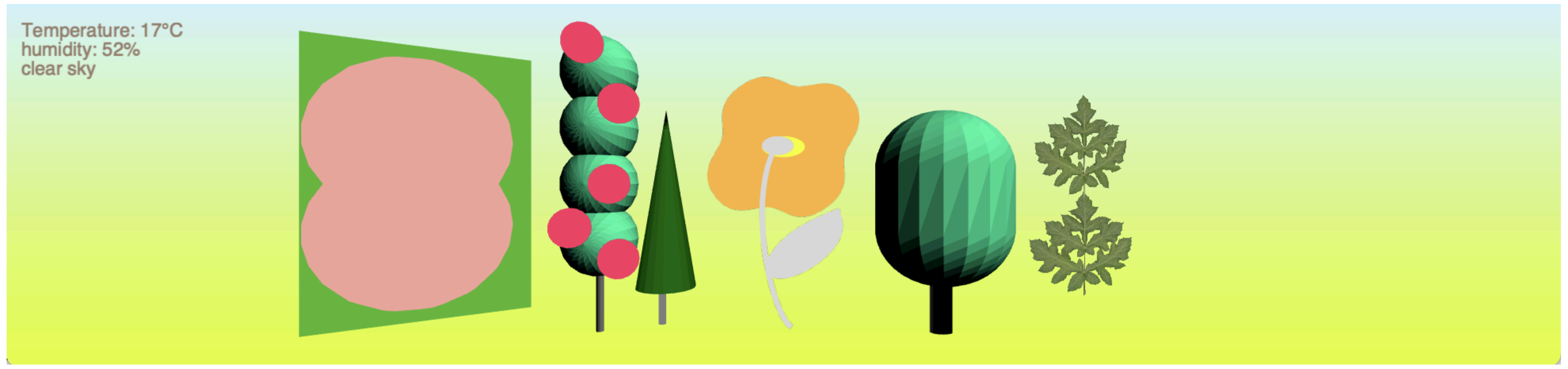
The deliverable

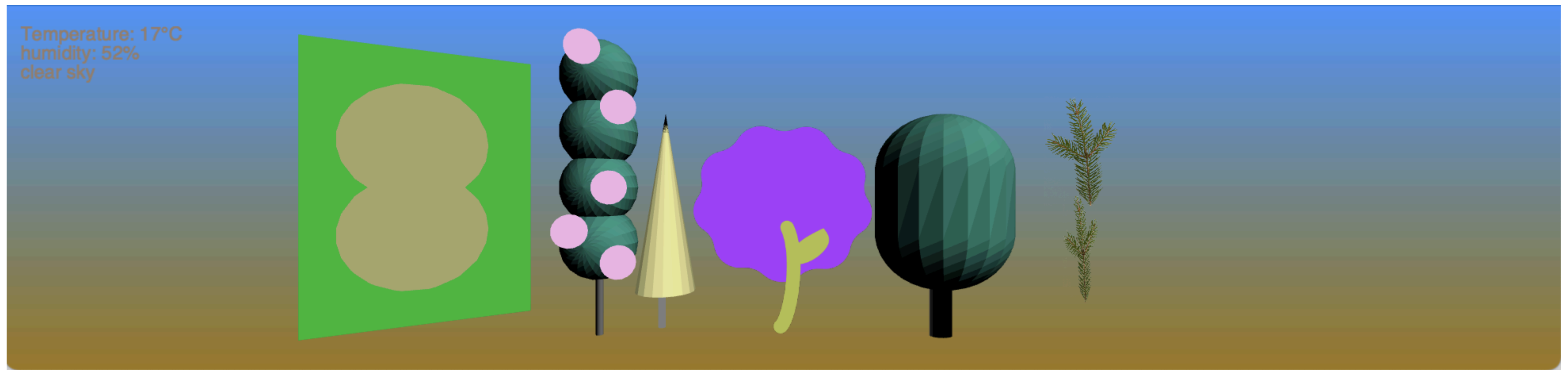
The project leverages Max/MSP for its real-time audio processing capabilities to create responsive visual elements. The system detects and transforms these aspects into visualizable parameters:

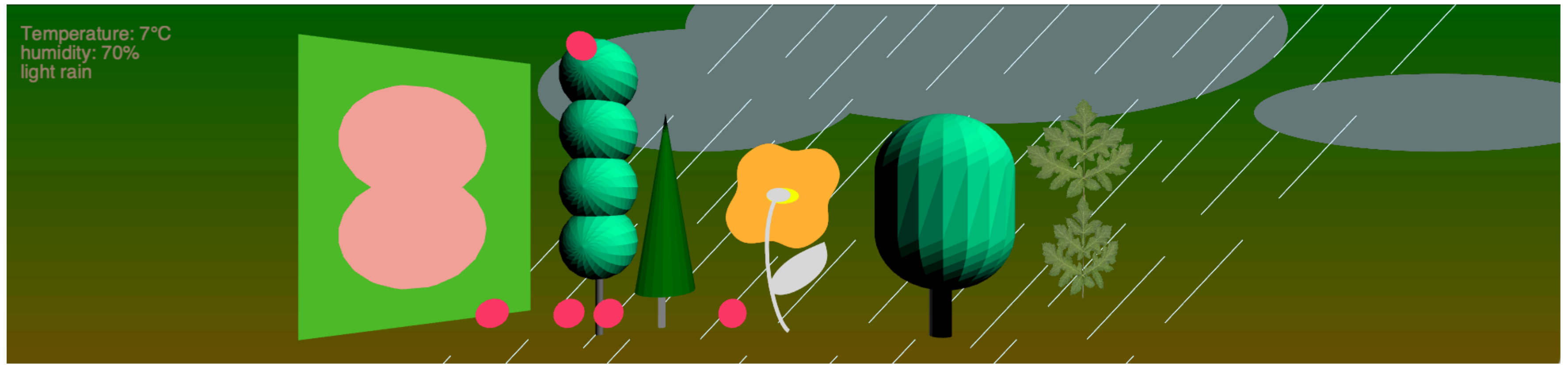
- Frequency spectrum analysis separates high frequency vocals (2kHz-8kHz) and low frequency bass (60Hz-250Hz) components, controlling element size and density
- Beat detection algorithms influencing rotational movement
- Harmonic analysis

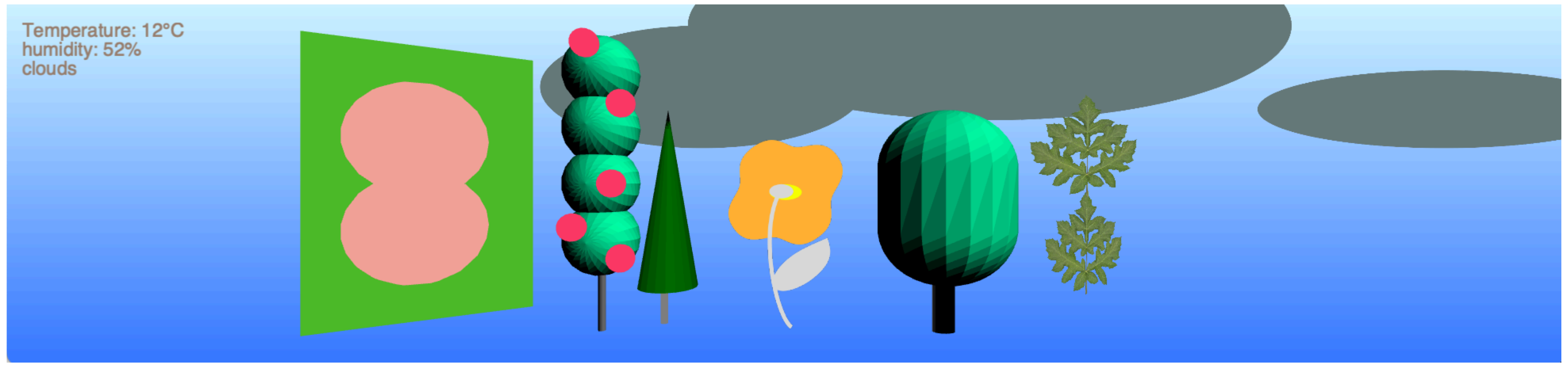
The visualization's atmosphere dynamically adapts to real-time weather conditions. During rainfall or overcast conditions, matching imageries will show in the background, where as on clear days, the background maintains a clean gradient.

To provide a minimalistic aesthetic, the different times throughout a day are shown through the dynamic color schemes.





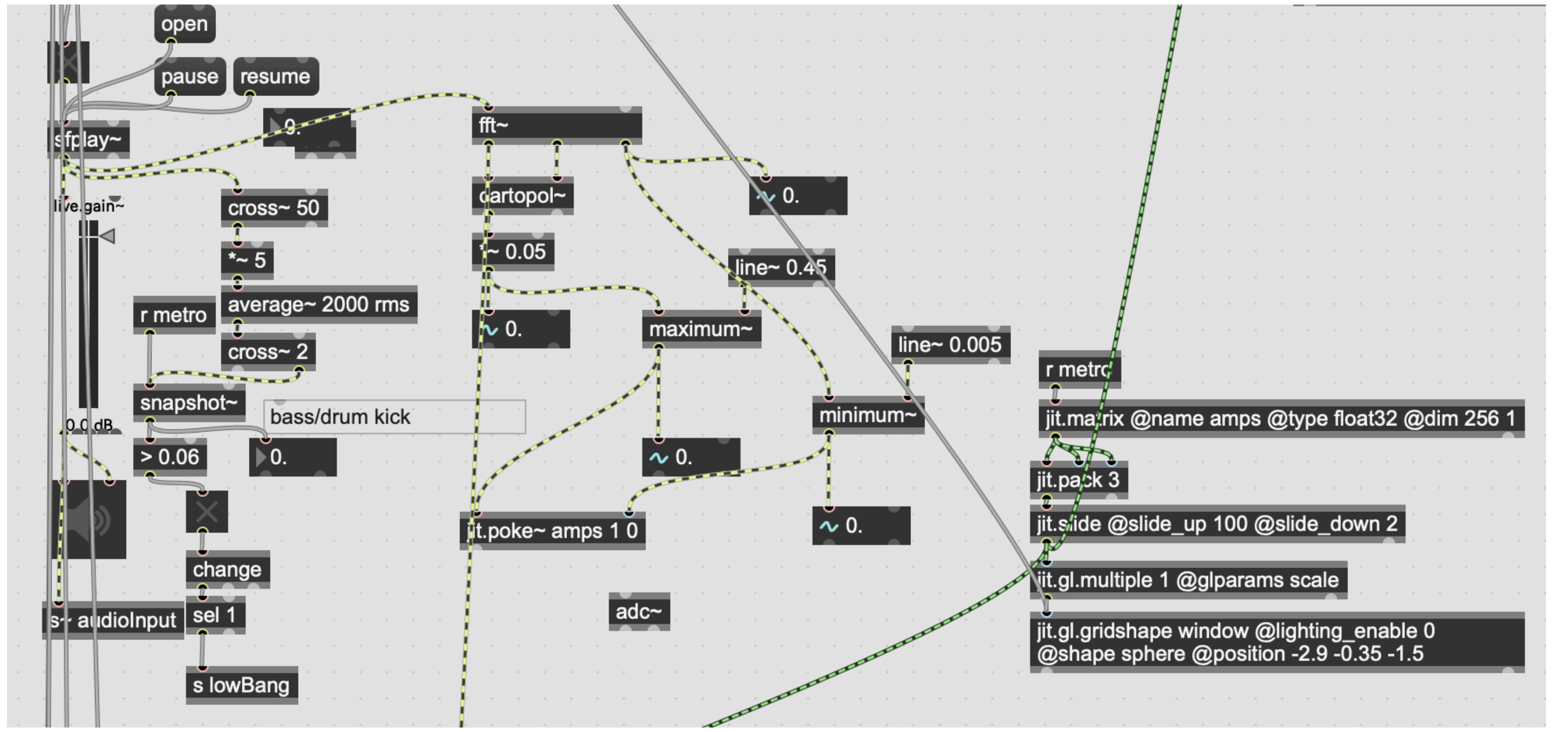




How it works

The max patch takes the time-domain audio input and applies fast fourier transform to obtain the frequency-domain data. The size of the graphic elements will be determined by a normalized and scaled amplitude of the respective frequency range.

The python script calls the OpenWeatherMap API and sends the weather data over a local port. Upon receival of this information, the patch parses the message and toggles the corresponding background elements.



FFT processing in the max patch

Python code snippet

```python
def send_to_patch(value, str, port):
    parser = argparse.ArgumentParser()
    parser.add_argument("--ip", default="127.0.0.1")
    parser.add_argument("--port", type=int, default=port)
    args = parser.parse_args()
    client = udp_client.SimpleUDPClient(args.ip, args.port)
    client.send_message(value, str)
```

Impact & Reflection

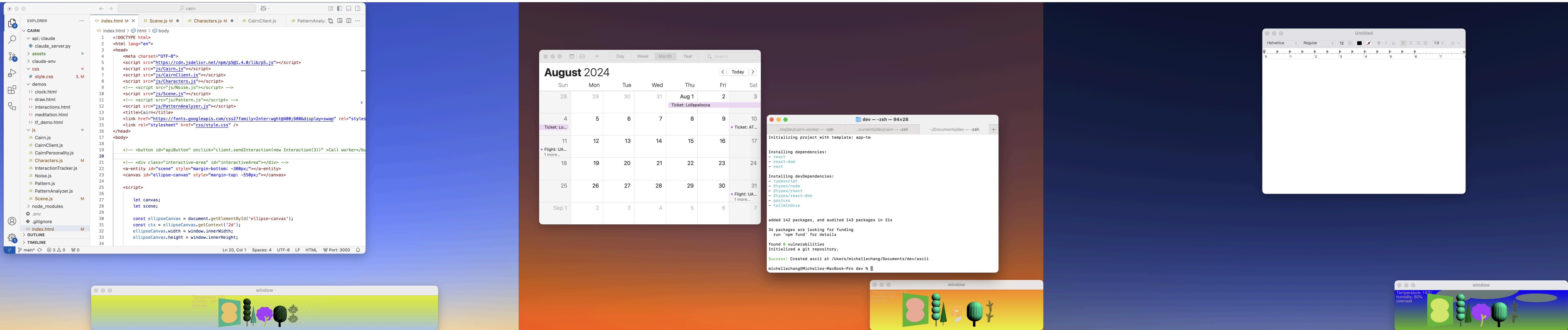
"I find myself glancing at it during code compilation or between tasks. It's become part of my work rhythm." - developer

"The subtle weather changes help me feel less disconnected during long meetings in our windowless conference room." - product manager

"I appreciate that it doesn't demand attention but adds ambiance. The time-based colors actually help me track my day better." - manager

The project revealed unexpected benefits in temporal awareness and workplace rhythm. By balancing minimal cognitive overhead and meaningful environmental context, the window demonstrates how thoughtful integration of natural elements can enhance our indoor-focused work environments.

Example usage at different times throughout the day



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script src="https://cdn.jsdelivr.net/npm/p5@1.4.0/lib/p5.js"></script>
    <script src="js/Cairn.js"></script>
    <script src="js/CairnClient.js"></script>
    <script src="js/Characters.js"></script>
    <!-- <script src="js/Noise.js"></script> -->
    <script src="js/Scene.js"></script>
    <!-- <script src="js/Pattern.js"></script> -->
    <script src="js/PatternAnalyzer.js"></script>
    <title>Cairn</title>
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600&display=swap" rel="styles
    <link rel="stylesheet" href="css/style.css" />
</head>
<body>

    <!-- <button id="apiButton" onclick="client.sendInteraction(new Interaction(3))" >Call worker</bu

    <!-- <div class="interactive-area" id="interactiveArea"></div> -->
    <a-entity id="scene" style="margin-bottom: -300px;"></a-entity>
    <canvas id="ellipse-canvas" style="margin-top: -550px;"></canvas>

    <script>

        let canvas;
        let scene;

        const ellipseCanvas = document.getElementById('ellipse-canvas');
        const ctx = ellipseCanvas.getContext('2d');
        ellipseCanvas.width = window.innerWidth;
        ellipseCanvas.height = window.innerHeight;
```

```
Initializing project with template: app-tw


Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- typescript
- @types/node
- @types/react
- @types/react-dom
- postcss
- tailwindcss


added 142 packages, and audited 143 packages in 21s

34 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Initialized a git repository.

Success! Created ascii at /Users/michellechang/Documents/dev/ascii

michellechang@Michelles-MacBook-Pro dev %
```