Catoptric Surface Control Interface

Master's Project Report

Logan Farrow
github.com/loganfarrow/masters-project

Advisors: Chamberlain, Ahrens, Gill
Date: 05-04-2025

Abstract

This report summarizes the development of a catoptric (mirror-based) surface control interface using Apple's SwiftUI framework. Over the course of a semester, a series of exploratory builds—ranging from basic SwiftUI tutorials to a full-fledged wireframe in React culminated in a proof-of-concept "remote control" for the Chamberlain, Ahrens, and Gill mirror array located in Steinberg Hall. The project demonstrates the advantages of SwiftUI over UIKit, outlines the architecture of both a React-based web wireframe and a native SwiftUI app, and proposes future enhancements including automated image processing, 3D visualization, and cross-platform support.

1. Introduction

Modern interactive lighting installations—such as the Steinberg Hall array of motorized mirrors—require intuitive control interfaces such that each single mirror needs to have fine-tuned control while groups of mirrors need to be moved as one. These controls can be used for a variety of services such as concentrated light for reading, as well as project art installations. Traditional iOS development has relied on UIKit and storyboards, which can become extremely cumbersome as projects grow due to the framework using a blend of block based coding and more traditional text based coding. With Apple's introduction of SwiftUI in 2019, developers gained a declarative, cross-platform UI framework offering live previews and state-driven updates. This project explores SwiftUI's capabilities through progressive app builds, applies those skills to a mirror-array control interface, and outlines next steps toward a production-grade system while deploying a functional demo to a real iPhone.

2. Learning and Comparing Frameworks

2.1 Swift and SwiftUI Fundamentals

Swift, introduced in 2014, is a modern language influenced by Rust, Haskell, and Python. SwiftUI, launched in 2019, is a framework built on Swift that uses declarative syntax to define UI, supports live previews, and automatically updates views when state changes. It unifies development across iOS, macOS, watchOS, and tvOS.

2.2 UIKit vs. SwiftUI

UIKit relies on storyboards and imperative wiring, which can lead to desynchronize and make future enhacements difficult. In SwiftUI, developers do not have to synchronize between many storyboards while mapping transitions by hand. Ever since SwiftUI's release Apple has been strongly pushing the full adoption the new framework. Most apps on the AppStore today use SwiftUI, but a major drawback is that it does not support older iOS versions. SwiftUI has support for including some elements of UIKit for more advanced functionality. SwiftUI's code-first approach (similar to HTML/CSS) removes storyboard drift, enables cross-platform reuse, and benefits from Apple's ongoing enhancements.

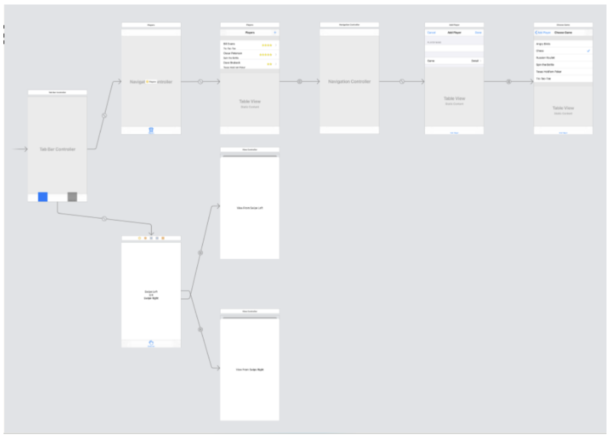

Figure 1: Old Storyboard Navigation Example

3. Development Approach

3.1 Learning Strategy

1. **Learn:** Studied Apple documentation, WWDC videos, and community tutorials on SwiftUI (Views, Modifiers, NavigationStack, state).

2. **Build:** Created sample apps to experiment with lists, navigation, networking (URLSession + Codable), drawing (GeometryReader + ZStack), and animation.

3. **Document:** Captured reusable patterns for the mirror-array control app. Built a long list of notes taking into consideration the best applicable ideas to the Mirror Control App.

3.2 Sample App Builds

App	Purpose	Takeaways
Sandwiches	Intro to images, lists, navigation	Image asset management; List + NavigationLink
3D Visualizer	RealityKit & ARKit exploration	AR session config; UIKit–SwiftUI bridging
National Parks	Grid layouts; server data	JSON decoding; complex view composition

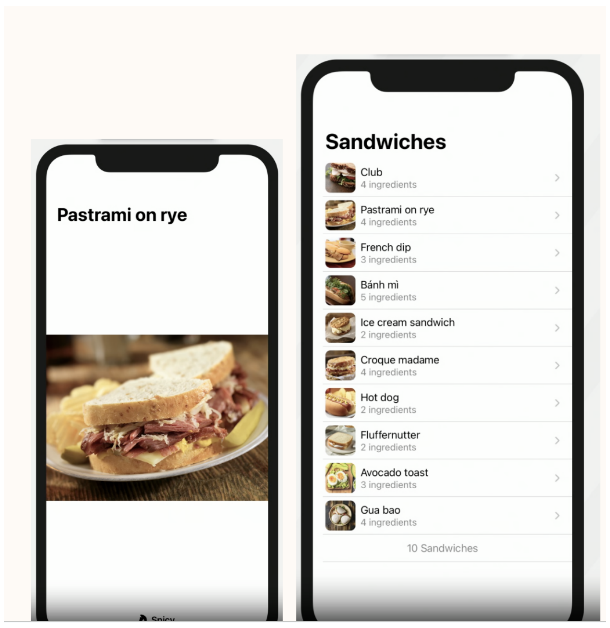

Figure 2: First Experiment - Sandwich App

4. Mirror-Array Control Interface

4.1 Web Wireframe (React)

A React prototype illustrated the desired workflow in detail and was presented to my advisor. Feedback on the wireframe was used to influence the final design to be adopted in the SwiftUI + Swift app.

- **Overview page:** Visual representation of the mirror array.
- **Sub-section control:** Fine-grained pan/tilt adjustment for mirror groups.
- **Image uploader:** Submit images for Grasshopper/Python processing.

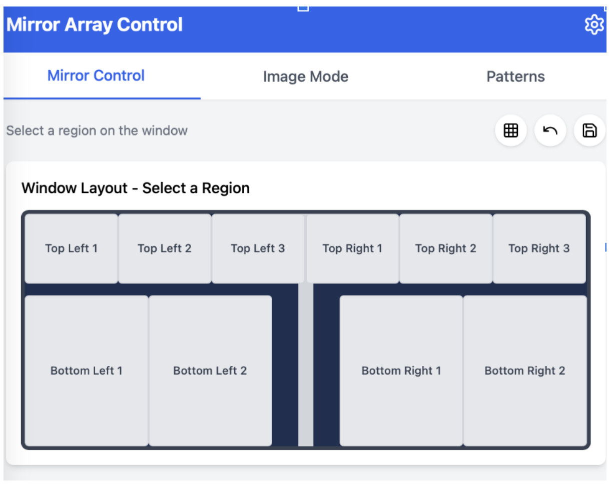

Figure 3: Wireframe running on React and Vite server.

4.2 Native SwiftUI Application

The SwiftUI app includes:

- **Dashboard view:** Dynamic grid of mirror segments via `LazyVGrid`. Each single mirror rendered as selectable button for user control.
- **Detail control:** `Slider` and `Stepper` bound to `@State` for pan/tilt. Allows dynamic selection of multiple mirrors to move in unision, as well as allowing user to select mirrors by section of the window.
- **Image uploader:** Allows users to edit their images live and see the changes made immediately. Images uploaded here would be sent to the middle-man server running the flask application to process images detailed in the future work section.
- **Chair-view lighting:** Custom UI directing light toward user's seat. As discussed in future work, the visualization can be improved using features from Swift's RealityKit and ARKit.

Images of each of the views and sections of the app can be found at the end of this report in figures three through five.

5. Conclusions and Future Work

5.1 Future Work

1. **Automated Image Processor:** Convert Grasshopper scripts to Python modules; host via Flask for image uploads and mirror orientation computation.
2. **3D Visualization:** Use RealityKit/ARKit (and VisionOS 3) for live 3D mirror-array models and spatial focal-point selection.
3. **Android Support:** Develop in Kotlin + Jetpack Compose; share HTTP command protocol with Flask backend.
4. **Web Support:** Connect the React wireframe to the server running the code for controlling the mirror array.

5.2 Conclusion

Using an iterative process I was able to successfully leverage my skills in React to develop a strong wireframe for the remote control application. After receiving feedback on this wireframe, I created three different sample apps to practice and learn SwiftUI. Once I had a strong grasp on SwiftUI, I developed a SwiftUI implementation of the Mirror Control App and deployed it to my personal iPhone. This project allowed me to experience designing a successful project from wireframe to deployment from the ground up.

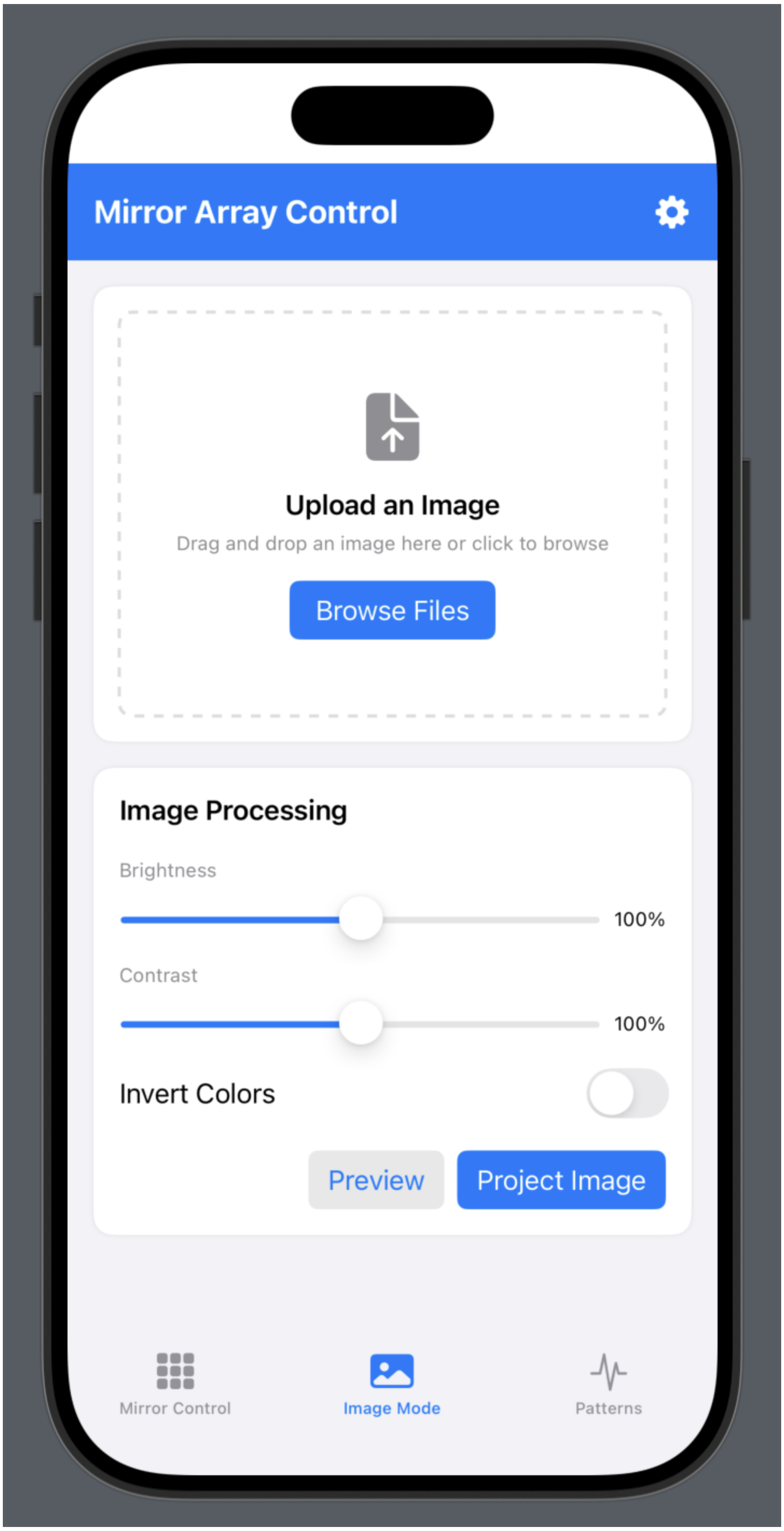

Figure 4: Image Uploading to Python/Grasshopper Script

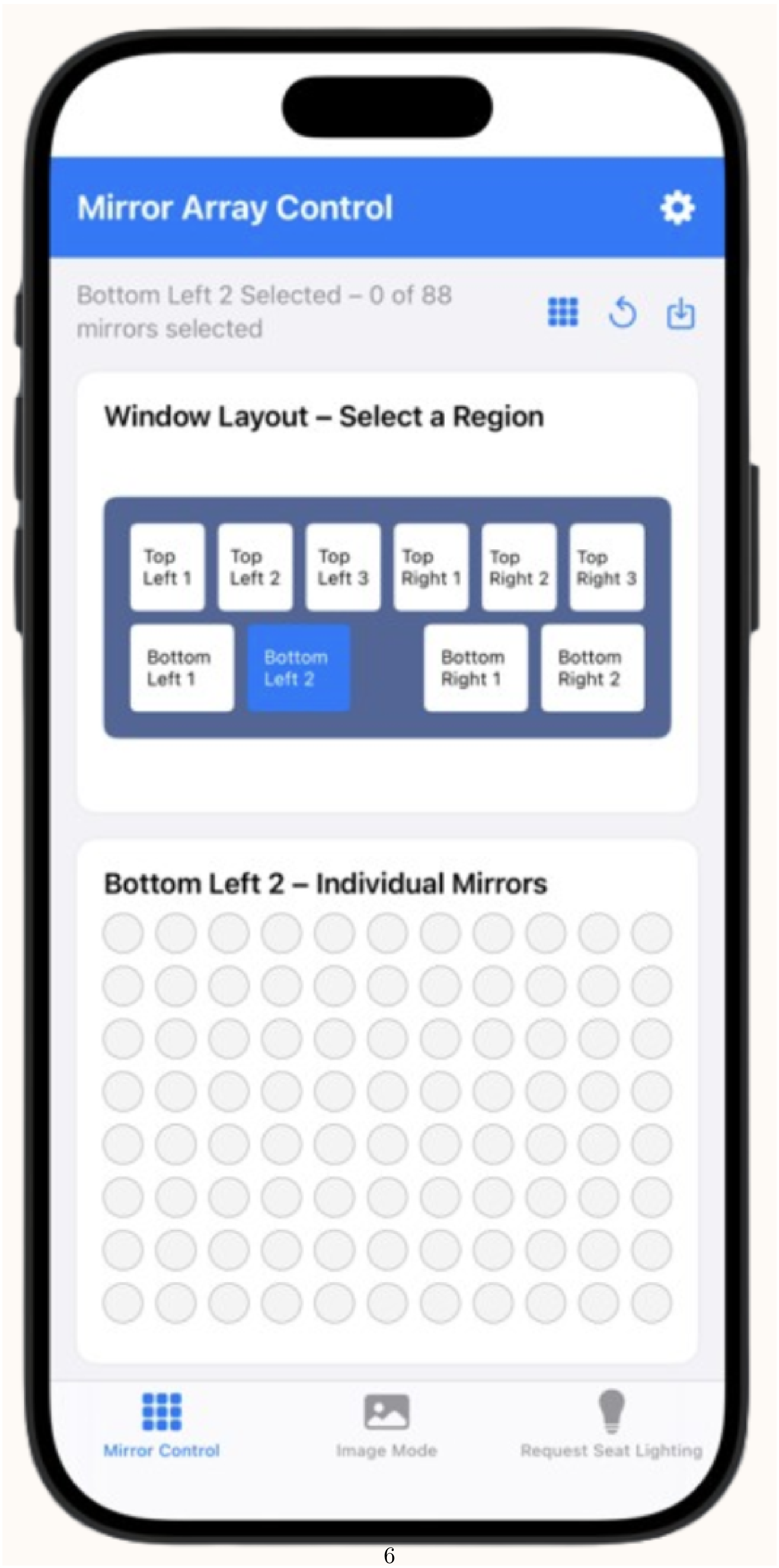

Figure 5: Mirror Control Interface

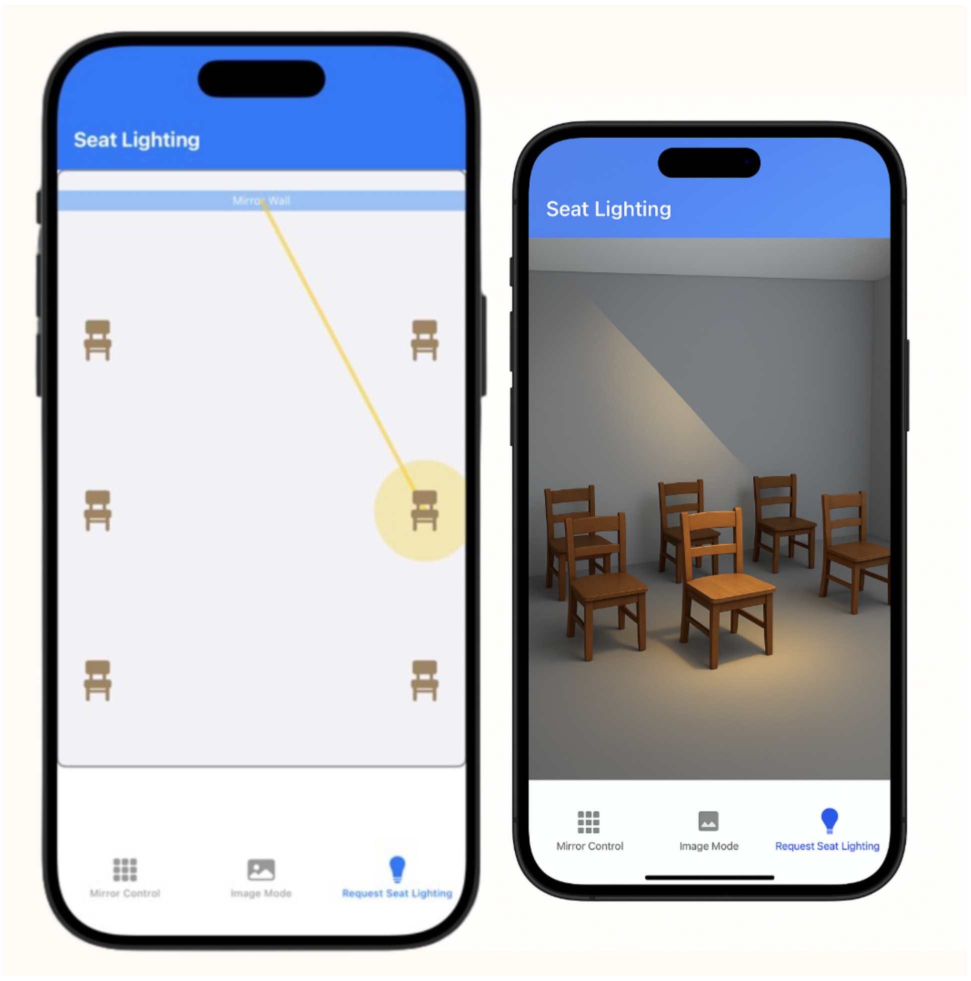

Figure 6: Seat lighting Control