VIRTUAL REALITY ENVIRONMENT FOR SMALL FISH

A Design Project Report

Presented To the School of Electrical and Computer Engineering of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Jack Strope

Meng Field Advisors: Van Hunter Adams, Bruce Land

Degree Date: May 2025

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: Virtual Reality Environment for Small Fish

Author: Jack Strope

Abstract:

This project supports research conducted by Dr. Andrew Bass and his team on *Danionella dracula* (“dracula fish”) by developing a virtual reality system for presenting visual stimuli to the fish. During the first semester, Python-based programs were written to display different visual stimuli on a projector using a Raspberry Pi. The second semester focused on building a user-friendly interface to allow researchers to run experiments independently. A touchscreen-based graphical user interface (GUI) was designed for launching programs and editing parameters directly from the PiTFT display. This eliminated the need for keyboard and terminal interaction, making the system accessible to researchers without backgrounds dealing with embedded operating systems. The final system allows for automatic startup, touchscreen-based control, and parameter editing, and is modular for easy future expansion. This report details the original goals, design changes, implementation challenges, and final outcomes.

Executive Summary

This project supports behavioral neuroscience research on *Danionella dracula*—a newly discovered, transparent fish species—by developing a virtual reality (VR) environment for delivering visual stimuli. Conducted in collaboration with Dr. Andrew Bass's lab in the Department of Neurobiology and Behavior at Cornell University, I worked on this project over two semesters and focused on both stimulus generation and system usability.

In the first phase, I wrote a series of Python programs to present visual patterns such as moving gratings and dynamic objects via HDMI to a projector, simulating ecologically relevant stimuli on the side of a fish tank. These programs allowed researchers to test the fish's perception of spatial and temporal motion. Originally, the plan for the second phase was for the first experiments to be performed over winter break, and for the researchers to decide on next steps (new programs, modifications to the programs, more advanced fish animation, etc.). However, due to delays in the team's research on other projects, the team's initial experiments on the dracula fish had to be postponed. Due to this change in plans, and my discussion with the researchers on the usability of the system, I changed the focus of the second phase to designing a user-friendly interface to facilitate the use of the system for their experiments.

To achieve this, a Raspberry Pi-based control system was enhanced with a PiTFT touchscreen interface. A custom GUI was developed using Pygame to allow program launching and parameter editing directly from the touchscreen, eliminating the need for keyboard or terminal access. This GUI includes multiple pages for launching stimulus programs and embedded JSON-based parameter editors for real-time experimental configuration. The system launches automatically at boot, making it user-friendly and robust for experimental use.

Background and Problem Statement

Dr. Adrew Bass and his research group in the Department of Neurobiology and Behavior at Cornell are conducting a new study on *Danionella dracula*, or "dracula fish," a relatively newly discovered species of tropical fish. These fish are particularly interesting for neurobehavioral research because their bodies are fully transparent, allowing their brains and other internal organs to be seen from the outside. Their brains and nervous systems are also semi-transparent, making them currently the only fish in the world whose brains can be fully mapped at single-neuron resolution purely from visual processing techniques. This makes the fish a compelling model for studying brain structure and function in vivo, and it has the potential to serve as a reference species for understanding sensory processing and behavior in other fish and vertebrates.

One of the research group's primary goals is to investigate how these fish perceive and respond to visual stimuli. Doing so provides a foundational understanding of how their sensory systems operate and lays the groundwork for investigating the neural mechanisms behind social behavior and communication. To carry out such studies, the team needed a system for displaying and controlling such visual stimuli. The goal of this project was to assist Dr. Bass and his team with their research by designing a projector-based virtual reality environment to present visual stimuli to the fish.

The experimental setup involves connecting a Raspberry Pi to a projector, which displays images generated by Python programs onto the side of a fish tank containing one or multiple fish. Initially, the project focused on building the Python stimulus programs themselves. Several of these programs were completed and tested by the end of the first semester. The original plan

was to then perform the first experiments during winter break and into the second semester and then proceed with new/modified programs depending on results, at the researchers' request. However, due to delays in the research schedule, the initial experiments had to be pushed back. Thus, during the second semester, I shifted the project toward usability. At this time, the Raspberry Pi required the user to type commands in the terminal to navigate to the project directory, launch the programs, and edit the programs' parameters by viewing the code in a text editor. Conversations with the research group led to the development of a PiTFT-based graphical user interface (GUI) for launching programs via a touchscreen. Later in the semester, further discussions prompted the addition of a parameter-editing GUI, allowing users to conveniently adjust experimental parameters without editing source code directly.

Although the system does not use immersive 3D hardware, it is effectively a form of virtual reality tailored to the sensory perception of fish. Unlike humans, whose forward-facing eyes provide stereoscopic depth perception, dracula fish have lateral eyes and limited binocular overlap. As a result, their perception of the world is largely two-dimensional. Thus, projecting dynamic 2D patterns—such as moving gratings or expanding circles—onto the side of the tank is sufficient to simulate environmental motion or visual cues. This VR-like method enables ecologically meaningful visual stimulation without the need for complex tracking or stereoscopic displays.

Planning

To design a system for displaying and controlling visual stimuli to the fish, a few design decisions had to be made early on. Dr. Bass and his team decided to use a projector displaying

images on the side of a small fish tank as the mechanism for delivering the visuals to the fish. It was also decided that a thin sheet of white paper would be affixed to the side of the tank. This diffuses the image, minimizing glare and improving visibility for the fish inside the tank. Another method that was considered was simply displaying the image on a monitor and placing the tank next to it. This would be a bit simpler, since the monitor could be placed a fixed (short) distance away and not require focusing like the projector does, but the projector/paper setup was ultimately decided on because it maximizes visibility of the image to the fish in the tank. Since the paper is attached directly to the side of the tank, and this is where the image is focused by the projector, this setup minimizes distortion compared to the monitor placed some small distance away. Furthermore, the way the image bleeds through the paper provides a much softer image for the fish, compared to directly displaying the output of the monitor/projector, which would likely produce too much glare and overstimulate the fish.

Another decision to make was how to generate the visuals to display to the fish. The simplest method considered was to simply display video playback of the desired stimulus. The problem with this method is lack of controllability. It's important that the researchers have a high degree of control over what is being displayed to the fish and can make slight adjustments to the stimulus to observe differences in the fish's behavior. An idea considered for displaying highly controllable images of fish was to use one of several fish animation programs designed by biologists to accurately mimic the movement of many species of fish. This would be useful for generating realistic fish animations to simulate a fake dracula fish. However, in the early stages of research on these fish, the researchers desired more simple visuals to display to the fish to obtain a baseline understanding of their visual systems.

Design and Implementation

Physical Setup

As described above, the mechanism for displaying visual stimuli is a projector displaying the image onto a sheet of white paper affixed to the side of a fish tank in which one or multiple fish will be located. The visuals are generated by Python programs running on a Raspberry Pi 4 model B and displayed on the projector via HDMI. During the second semester, a PiTFT touchscreen was added to the Raspberry Pi, featuring a 320x240 touchscreen display and four buttons on the side connecting to GPIO pins. Figure 1 below shows an example of the final experimental setup. The image depicts one of the programs displaying a grating acuity test to the tank.

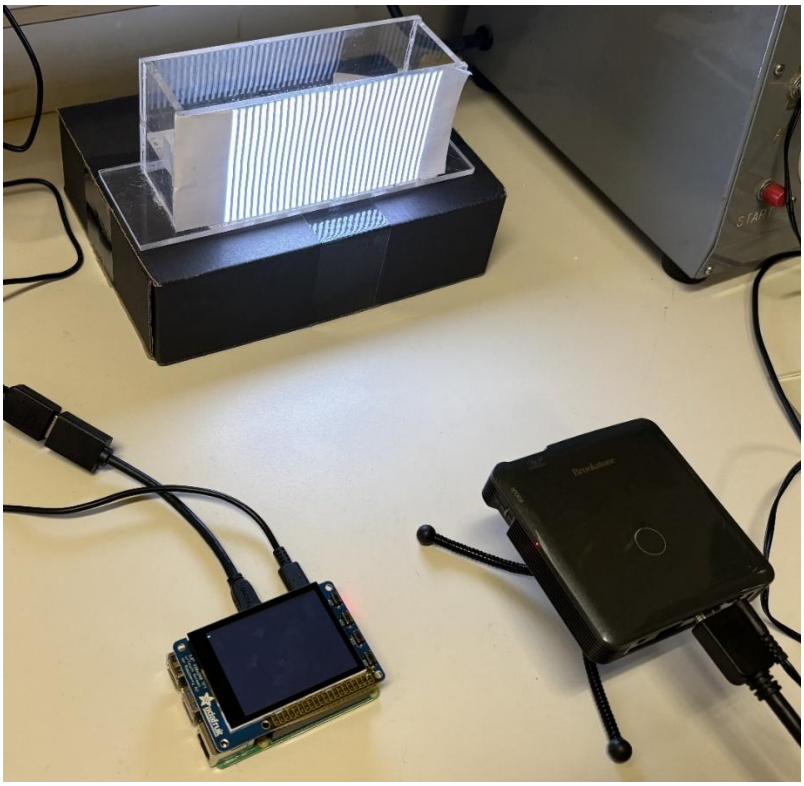

Figure 1: Physical setup displaying a grating acuity test

Visual Stimuli Development

The initial phase of the project involved designing and implementing the Python-based programs responsible for displaying visual stimuli to the fish. These stimuli were designed using the Pygame library, which offered simple yet flexible tools for rendering graphics directly to the HDMI-connected projector from the Raspberry Pi. Each stimulus was written as a standalone Python script and focused on presenting a specific pattern or behavior, such as:

- **Grating acuity test:** A pattern consisting of black and white bars which oscillate back and forth to measure the fish's sensitivity to spatial and temporal frequencies.
- **White circle stimulus:** A white dot that moves to elicit directional attention or approach/avoidance behavior.
- **Mouse-control stimulus:** A moving circle that tracks the position of the mouse, allowing researchers to test dynamic motion tracking.
- **Boids-based motion:** An implementation of the classic Boids flocking algorithm that simulates the motion of multiple "fish" on-screen, with potential to test social behavior responses.

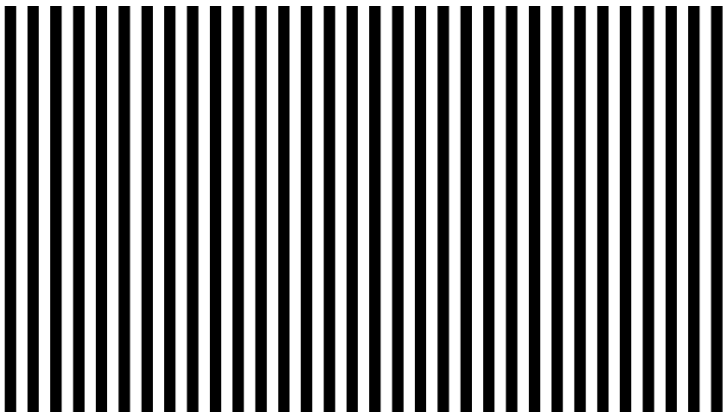

Figure 2: Grating Acuity Test

Each program was designed to run full-screen and loop continuously until a GPIO-based exit button was pressed. Each program has a set of parameters that can be adjusted to modify program behavior. For example, for the grating acuity test, the width of the bars, the speed at which they move, the distance they move before switching direction, and the angle at which they oscillate are adjustable parameters. Initially, parameters could only be adjusted by editing the source code, so they were placed near the top of the program with comments indicating where to change the values.

GUI Development

During the second semester, focus shifted to system usability. Since researchers were not experienced with Linux or Raspberry Pi systems, relying on a terminal to launch and modify programs presented a barrier. So, to facilitate user interaction, a PiTFT touchscreen was added to the system, and I developed a simple graphical user interface (GUI) to allow programs to be launched by simply pressing buttons on the screen. To automatically launch the GUI on startup, I created a shell script to set the environment variables for the PiTFT, projector, and mouse (if one is connected for the mouse-control program) and launch the GUI. I then created a system service which configures the Raspberry Pi to run this script when it boots up. This prevents the user from needing to plug in a mouse and keyboard to the Raspberry Pi to use the system, since it automatically enters the GUI on startup, which the user can navigate using only the touchscreen.

The GUI was implemented as another Python program, `fish-ui.py`—this is the program that is launched on startup by the shell script. The GUI was organized into multiple

pages, each containing buttons for launching a subset of available programs. Buttons are displayed using Pygame to print text to the screen and to react to touch events at the location of the button. Touch input was handled through Pygame's MOUSEBUTTONUP events, which detect when a mouse button is let go after being pressed. This works because touch events on the PiTFT are treated as mouse button clicks—tapping the screen at a given location on the screen is treated as a left-click with the mouse with the cursor at that location. The GUI's pages are managed by storing the current page as a Python enumeration. The program loops at 60 frames per second, and for each loop it checks the current page and prints the corresponding buttons. Navigation buttons are printed at the bottom of each page to allow the user to switch pages or exit the GUI. When one of these buttons is pressed, the current page is simply switched to the page corresponding to the button pressed, so that the new page will start being printed on the next loop iteration. When a program launch button is pressed, the GUI uses `subprocess.call()` to run the corresponding Python program. Each program is configured with an exit callback function, which exits the program back to the GUI when the bottom button (connected to GPIO 27) on the PiTFT is pressed.

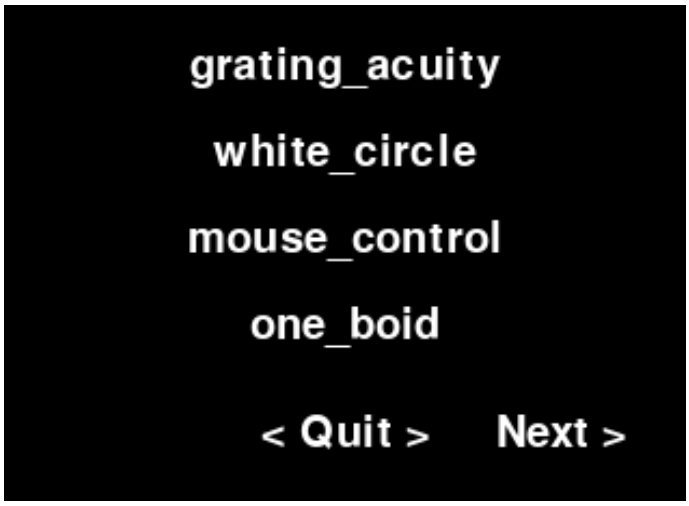

Figure 3: Example page of the GUI

After implementing the main GUI, one final request made by the researchers was to implement a way to easily adjust parameters between runs of the visual stimuli. They wanted to be able to run a program, stop it to slightly adjust a parameter, and then rerun the program immediately, since this best follows their scientific research workflow. To implement this, I added a new GUI for each program, in the same style as the main one, that could be launched while running the program. To allow parameters to be adjusted programmatically by the parameter editors, each program's parameters were moved to a JSON file, so that each program now loads these as its parameters before running the program. The parameter editor displays the current values of each of the program's parameters, along with plus and minus buttons next to them. The user can press the plus/minus buttons to increase and decrease the parameters by set increments. With the buttons at the bottom of the screen, the user can then save the current parameters and run the program again, or cancel to revert the parameters to what they were before and exit back to the main GUI.

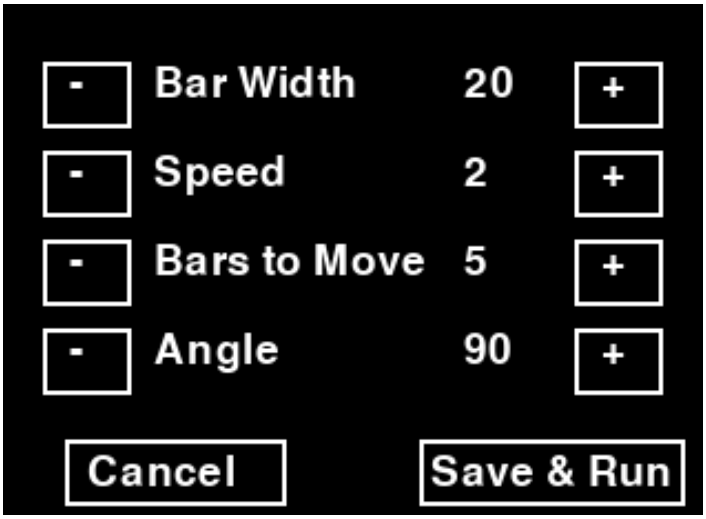

Figure 4: Example parameter editor for grating acuity test

To support a seamless user experience, the system uses a structured control flow to transition between the main GUI, the parameter editor, and each visual stimulus program. When a program is launched from the main GUI, the user can either terminate it as usual to return to the main GUI by pressing the bottom button (GPIO 27) or enter the parameter editor by pressing the top button (GPIO 17). Internally, this distinction is implemented by having each stimulus program return a specific exit code: 0 for normal termination and 10 when the user requests parameter editing. 10 was chosen to prevent conflicts with other common exit codes—generally, nonzero exit codes are used to indicate the program terminated with an error, but here, it's used to communicate to the main GUI what the user wants it to do. This exit code is returned by `subprocess.call()` to the main GUI. The GUI checks this code after the program finishes and, if it receives the code 10, launches the corresponding parameter editor GUI. Otherwise, it proceeds as normal and waits for the next action from the user. This control flow is visually represented in Figure 5 below.

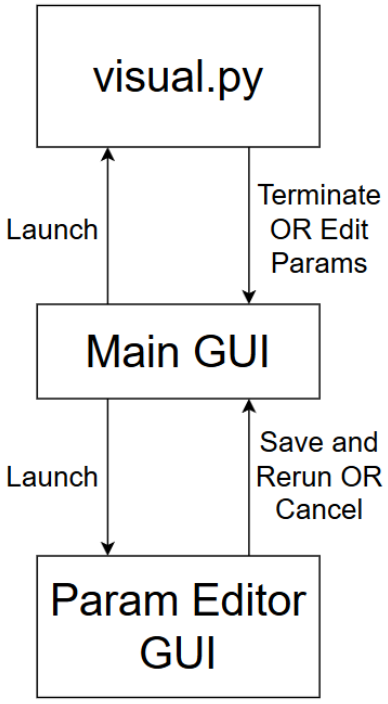

Figure 5: Diagram visualizing the GUI control flow

Testing and Results

The final system met all functionality goals and was tested across multiple runs and configurations. All programs have been verified to launch correctly from the GUI, whether it was started automatically when the Pi boots up or launched manually from the terminal. All programs display at full resolution on the projector and exit cleanly using GPIO input. One issue I encountered when testing was that, although everything worked when launching the GUI from the terminal (as I mostly did while testing), the GUI struggled to give control of Pygame to the visual programs, preventing them from displaying and exiting properly. The issue turned out to be that the environment variables, which allow programs to interact with devices connected to

the Raspberry Pi, were not being set properly. To fix this, I added lines to the shell script to explicitly set up the environment variables before launching the main GUI.

After adding the parameter editor GUIs, I tested them extensively to ensure that every time, whether the main GUI was launched at startup or via the terminal, the parameters are properly adjusted, and either saved or reverted depending on which button is pressed to exit it. The visual stimuli are all displayed to the tank clearly with no perceptible flicker.

The original goal of this project was to assist Dr. Bass and his team with their research on dracula fish. This was successful, as the project provided them with a useful tool for presenting controllable visual stimuli to the fish for their experiments. The system is robust, so long as the physical configuration is properly set up before use—one caveat is that the projector must be plugged into the Raspberry Pi before powering it up to ensure the auto-start shell script can properly set the environment variables before launching the main GUI.

Future Work

With the completion of the stimulus delivery and control system, Dr. Bass and his team are now positioned to begin behavioral experiments on dracula fish using the VR environment developed in this project. Initial studies will likely focus on assessing the fish's responses to motion, contrast, and directionality through tests such as grating acuity and object tracking. These experiments aim to characterize the fish's visual processing capabilities and lay the groundwork for exploring the neural mechanisms behind social behaviors and communication.

Looking forward, future engineers supporting this project may extend the system in several ways. The code is all stored locally on the Raspberry Pi's micro-SD card was designed to be modular, allowing additional programs to be added to the project directory and launched via the GUI, using this report as a user's manual to learn how it works. To add a visual program, for example, the steps would be to add the program to the project directory (`/home/pi/fish`), make sure it uses GPIO 27 to exit with code 0, add a corresponding button to the main GUI to launch it, and add a JSON file to store the parameters if convenient parameter editing between runs is desired.

Conclusion

This project successfully delivered a robust and user-friendly virtual reality system to support visual behavior research on *Danionella dracula*. Through the development of custom Python programs, a touchscreen-based GUI, and startup automation, the system meets the researchers' needs for experimental flexibility and ease of use. It allows for precise control over visual stimuli and seamless parameter tuning between trials—capabilities essential for ongoing behavioral studies.

The modular design ensures future extensibility by other engineers or researchers, making it possible to expand the system without modifying its core structure. Overall, the system serves as a practical and accessible tool that enables the Bass Lab to begin the next stage of their research on sensory perception and neural processing in a unique and promising model species.

References

[1] Pygame Documentation. https://www.pygame.org/docs/

[2] Python Software Foundation. *Python subprocess module documentation.* *https://docs.python.org/3/library/subprocess.html*

[3] Cornell University. *ECE 5725 Embedded Operating Systems Laboratory Materials*, Fall 2024.

[4] Pita, Diana & Moore, Bret & Tyrrell, Luke & Fernández-Juricic, Esteban. (2015). Vision in two cyprinid fish: Implications for collective behavior. PeerJ. 3. e1113. 10.7717/peerj.1113.