Music Engineering Independent Project
May 6, 2025
Final Report

“Beats Hero” Rhythm Game

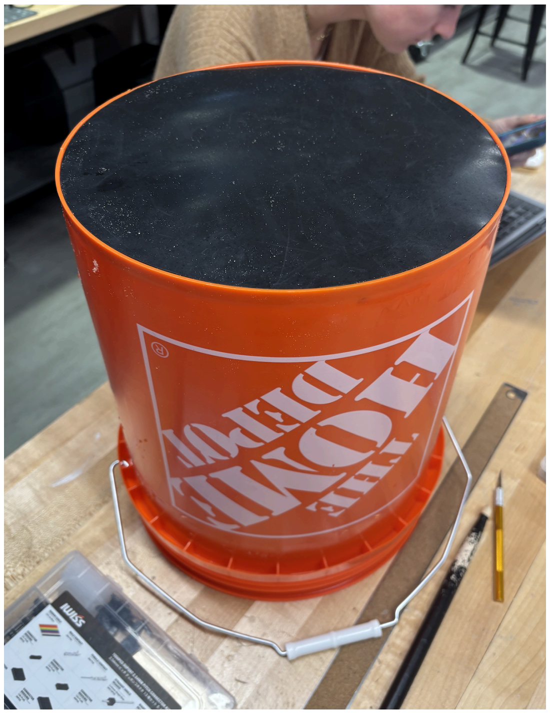
By Drew Cohen

I. Overview

For this project, I set out to recreate a type of “guitar hero” video game inspired by Tufts BEATs, a street drumming club on campus. The game involves a modified Home Depot bucket with two sensor areas corresponding to the main hits of the instrument, an on-bucket computer to process sensor data, and the game program coded in Godot that runs either off-bucket (for better performance) or on a Raspberry Pi.

II. Mechanical Assembly

The mechanical aspects of this project are relatively simple. The Home Depot bucket frames the instrument and houses the internal electronics. The top surface was smoothed with an orbital sander, and a 1/8” EPDM rubber pad was cut and glued over the surface sensors to improve the playing surface, similar to a practice pad.



III. Sensors and Circuitry

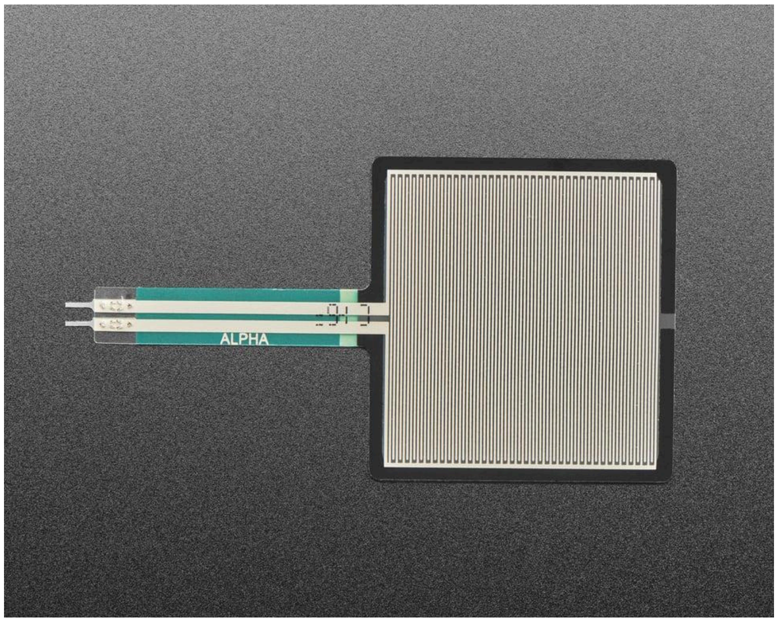


Figure 2: P1075 Square FSR

The sensing for this project utilizes two arrays of FSR banks at the center and rim of the pad. One 2x2 array of sensors is placed directly underneath the pad, detecting center hits, while two FSRs fold over the rim and detect rimshots. The output from these sensors is wired through a voltage divider and the ADC on the on-board computer

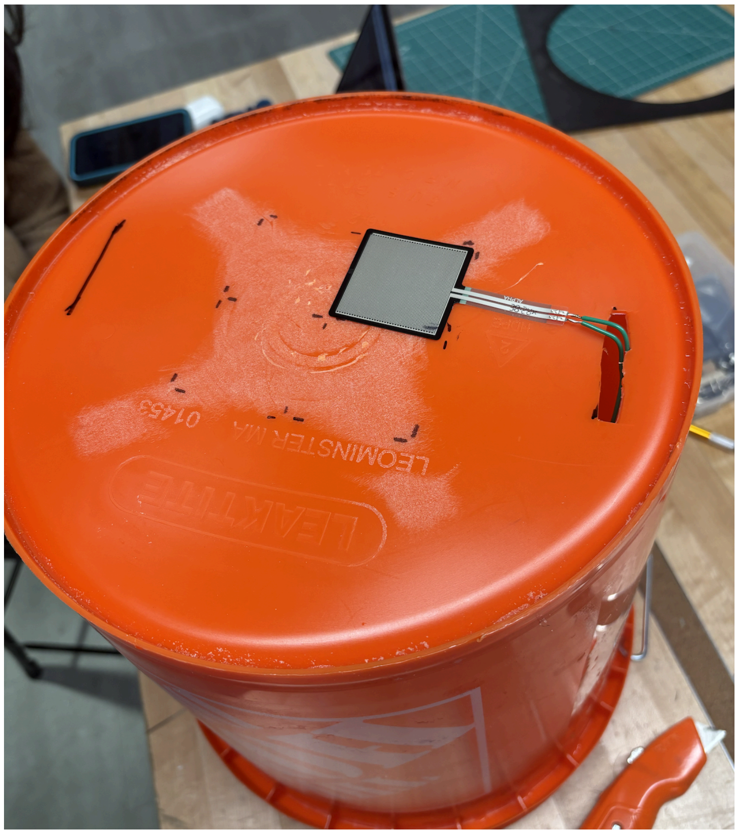


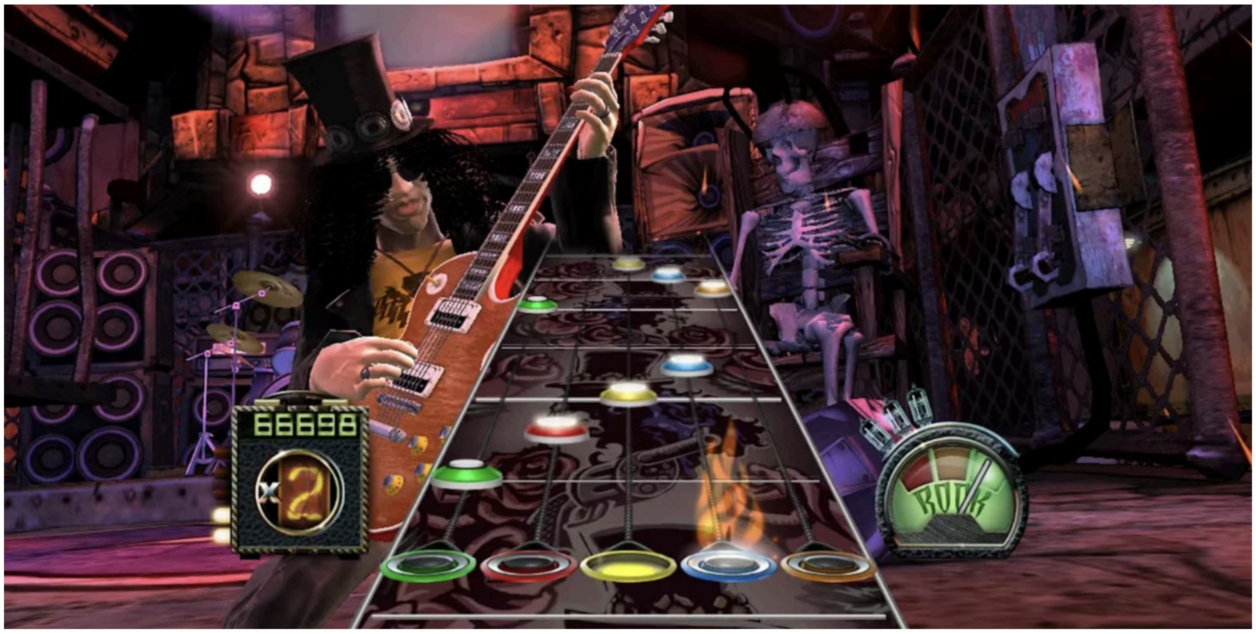
Figure 3: Example sensor under the rubber

IV. Software

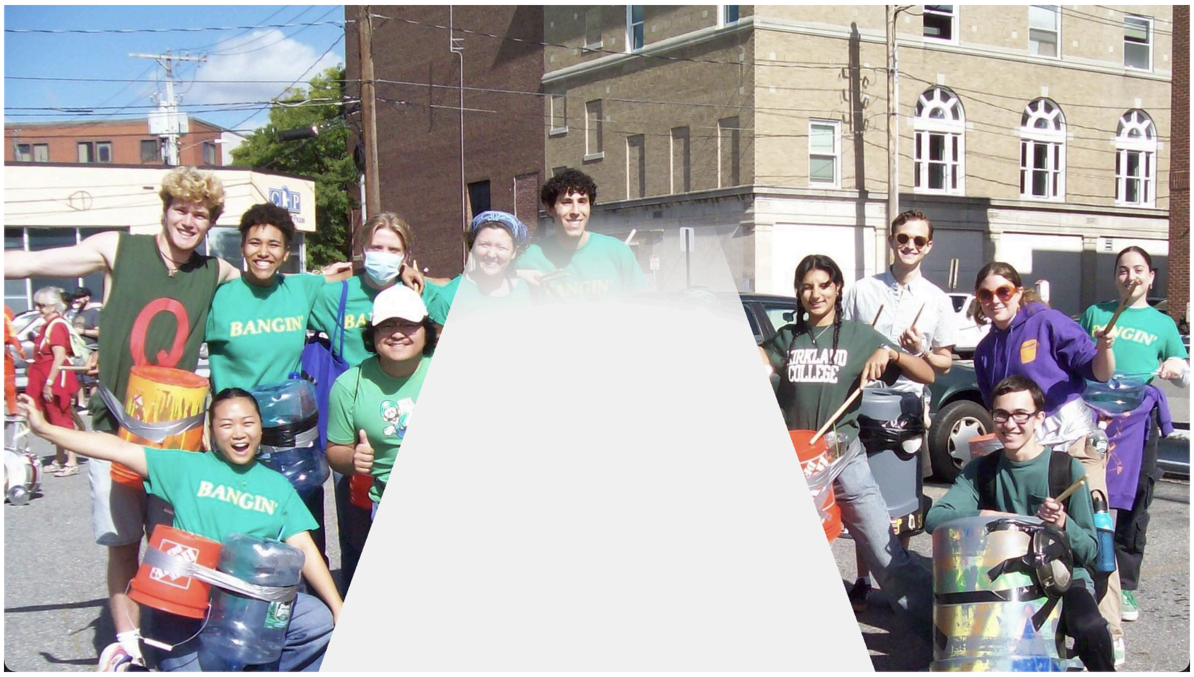
The software side of this project involved reading from the sensors, and creating a game to display the output and challenge the player to match hits with a song. It was by far the most involved aspect of the project, and went through several phases. Troubleshooting was such a large part of this process that this section would be incomplete without noting work that ultimately did not end up in the final game.

A. Godot Development

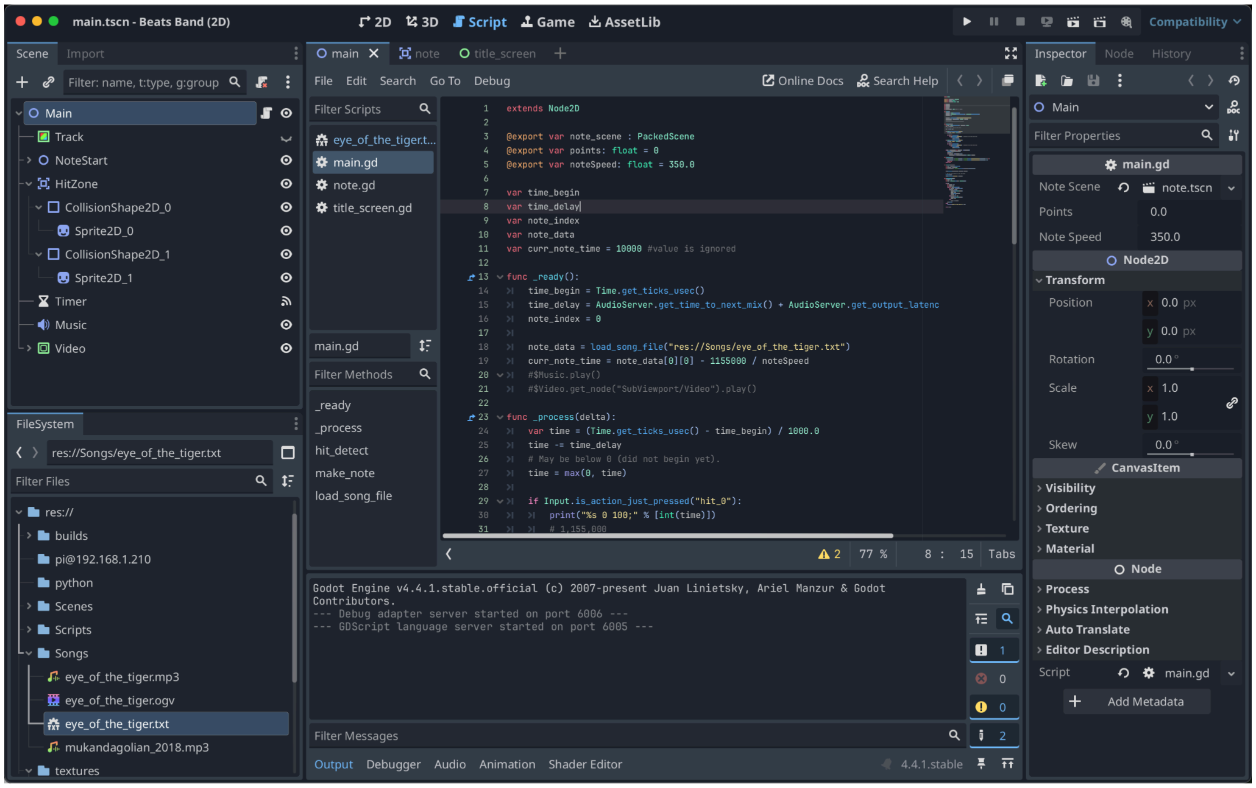
I chose Godot, a free and open source game engine, to write the game itself. I had developed games in general programming languages before but never with a dedicated engine, so enjoyed the challenge of learning a new skill. I began by trying to emulate the look of Guitar Hero, which looks like this:



I experimented with a 3D track and background image, and spent a few hours writing a custom shader that made objects more transparent the farther away they were.



At this point, I decided to move to a 2D game to save time. As it stands, the demo Node tree (Godot's abstraction is based around Nodes and Scenes, which are basically objects and classes) looks like this:

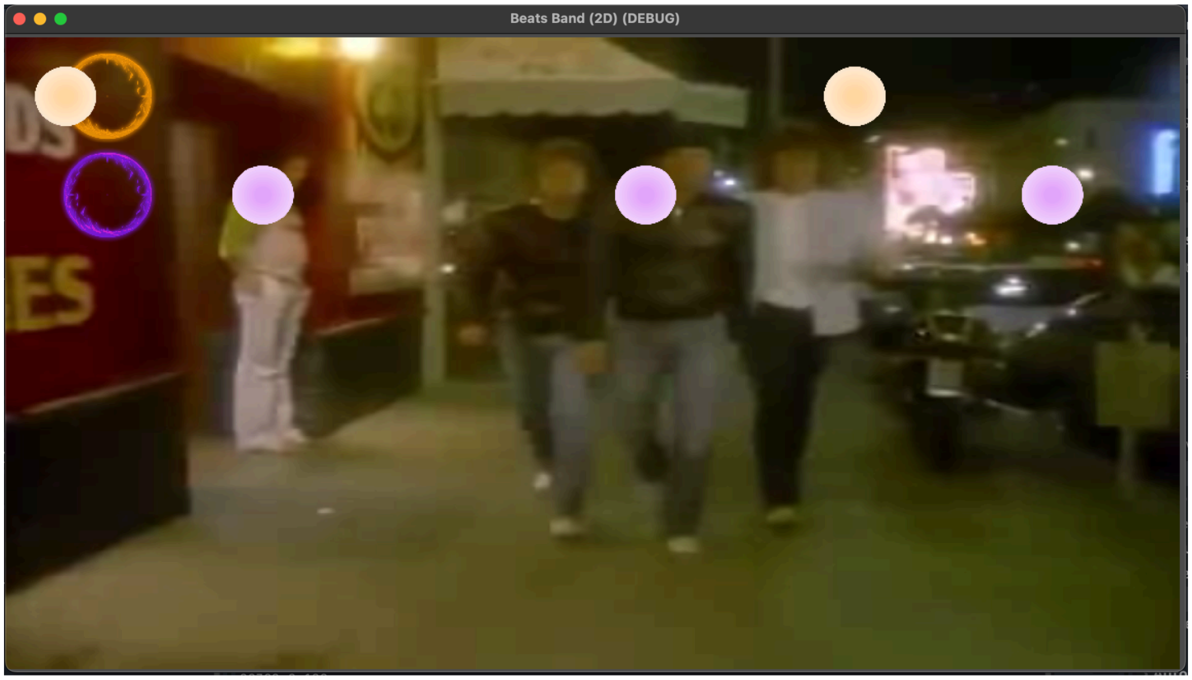


```
extends Node2D

@export var note_scene : PackedScene
@export var points: float = 0
@export var noteSpeed: float = 350.0

var time_begin
var time_delay
var note_index
var note_data
var curr_note_time = 10000 #value is ignored

func _ready():
	time_begin = Time.get_ticks_usec()
	time_delay = AudioServer.get_time_to_next_mix() + AudioServer.get_output_latenc
	note_index = 0
	
	note_data = load_song_file("res://Songs/eye_of_the_tiger.txt")
	curr_note_time = note_data[0][0] - 1155000 / noteSpeed
	#$Music.play()
	#$Video.get_node("SubViewport/Video").play()

func _process(delta):
	var time = (Time.get_ticks_usec() - time_begin) / 1000.0
	time -= time_delay
	# May be below 0 (did not begin yet).
	time = max(0, time)
	
	if Input.is_action_just_pressed("hit_0"):
		print("%s 0 100;" % [int(time)])
		# 1,155,000
```

```
Godot Engine v4.4.1.stable.official (c) 2007-present Juan Linietsky, Ariel Manzur & Godot
Contributors.
--- Debug adapter server started on port 6006 ---
--- GDScript language server started on port 6005 ---
```

Main
main.gd
Note Scene: note.tscn
Points: 0.0
Note Speed: 350.0
Node2D
Transform
Position: x 0.0 px, y 0.0 px
Rotation: 0.0 °
Scale: x 1.0, y 1.0
Skew: 0.0 °
CanvasItem
Visibility
Ordering
Texture
Material
Node
Process
Physics Interpolation
Auto Translate
Editor Description
Script: main.gd
Add Metadata

Note objects appear on the Track and move left until they enter the two collision areas in the HitZone. At this point, if the proper hit is registered, the Notes are considered to be ‘hit’ and disappear. A music video plays in the background (this test song is Eye of the Tiger):



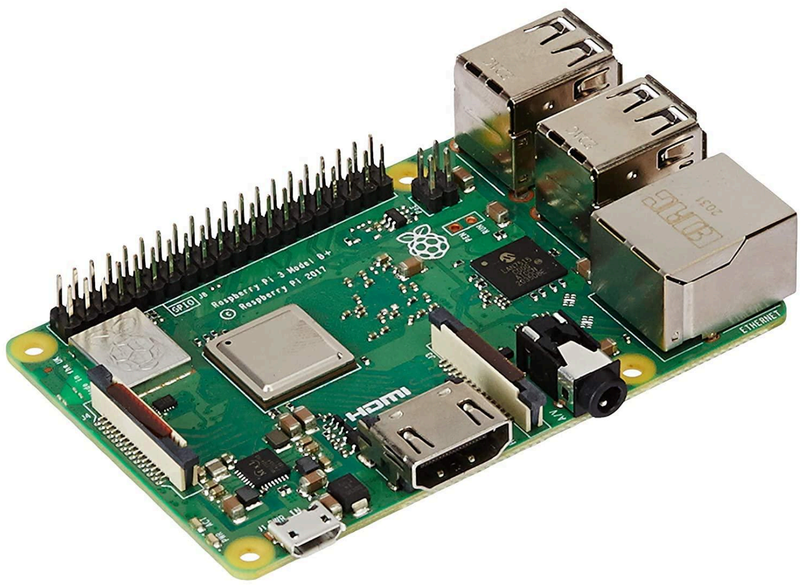
B. Song Storage

Like Guitar Hero, I needed to store songs in the game that users could choose and attempt to play. However, the only recordings of BEATs songs are live, with no track or constant BPM. This means the location of the notes must be stored individually in ms, not as unit time, and had to either be pulled from the recordings themselves or written manually.

To write songs (or add any new songs to the game), I added a recording mode. When you play a song in recording mode, it doesn't send any notes along the track but plays the .mp4 and records the timing of each user hit to a file. In normal mode, the game reads from this song file and sends notes along the track as recorded. Songs can thus be revised by future BEATs members as they see fit.

C. Raspberry Pi Integration

My vision for this project was a seamless plug-and-play, which required a powerful on-board processor to read notes, process the game, and output to HDMI.



The Raspberry Pi 3b+ has all of this functionality, so was ideal for this project. I had not used a Raspberry Pi from scratch, so it was exciting to learn how to load the operating system and set up the SSH. Preemptive research suggested that it would be difficult, but possible, to run Godot games on its hardware. Godot optimizes game projects to different operating systems and doesn't natively support Raspbian, an arm64

Linux distribution, but you can find templates on Github. Then, installing the Vulkan rendering driver allows any modern Raspberry Pi to run Godot games over HDMI.

The 3b+ input pins can be read over the command line, which Godot can access. In this way, user input is registered from the sensing circuit as described in the above sections. Alternatively, the game can be run on a laptop, integrated with user input over the serial port with an Arduino as is taught in EMID.

IV. Conclusion

Ultimately, I am really excited about the state of this game. I have a working drum pad on the bucket, a nice Godot foundation, and integration between the two. There’s a lot of work to continue beautifying the bucket and the game’s UI, as well as adding a scoring mechanism and more songs.

Unfortunately, the performance issues with the Raspberry Pi make it almost unplayable. I am still working on ways to optimize the game so that the plug-and-play vision can become reality – the hardware is not the issue, as it can stream .mp4 and surf the web without much stutter. As of now, I have an Arduino setup similar to EMID with the game reading off the serial port through a python script. It’s a lot of fun!