By Tom Shaw

Introduction and about me

I’m Tom. I’m a multiskilled electronics engineer with interests in AI & ML, ICs and electronics, VFX, art, maths & statistics, computer science and product development.

I have been using CAD tools since I was 12, my favourite film is Le Mans’ 66, and I am teaching myself to play piano.

In this portfolio I present a collection of my projects over the last few years. These are ones which posed significant challenges to me and have a story to tell, they span a wide range of topics, and I took a lot of pride in each one.

Enjoy!

Contents

Projects are in chronological order (most recent first). Project name, [Keywords/Topics/Software]

1. **Multi-agent formation control of UGVs (2 pages),** [Multi-agent controller, distributed cooperation, formation keeping, Sliding Mode Control, UGVs, graph theory, ROS/Gazebo, MATLAB/Simulink, LaTeX]
2. **PCB Art – Image2Kicad,** [Python, PCB, algorithms, graph theory, tracing, GUI, art]
3. **PCB design practice,** [4-layer PCB, motor driver, high voltage circuit, power electronics]
4. **The Magic Microphone,** [Image generation, Generative AI, Raspberry Pi, thermal printing]
5. **GAIT short film**, [Generative AI Tools (GAIT), Video generation, short film, 3D, Blender]
6. **Custom Mechanical Keyboard**, [CAD, PCB design, prototyping, Continuous refinement]
7. **Keyboard Showcase**, [Short film, CAD, animation, 3D, Blender]
8. **Nodevember 2020**, [Node-based programming, vector maths, shaders, 3D art, VFX, Blender]

Multi-agent formation control of UGVs (part I)

3rd year individual project supervised by Dr Allahyar Montazeri (August 2024-March 2025)

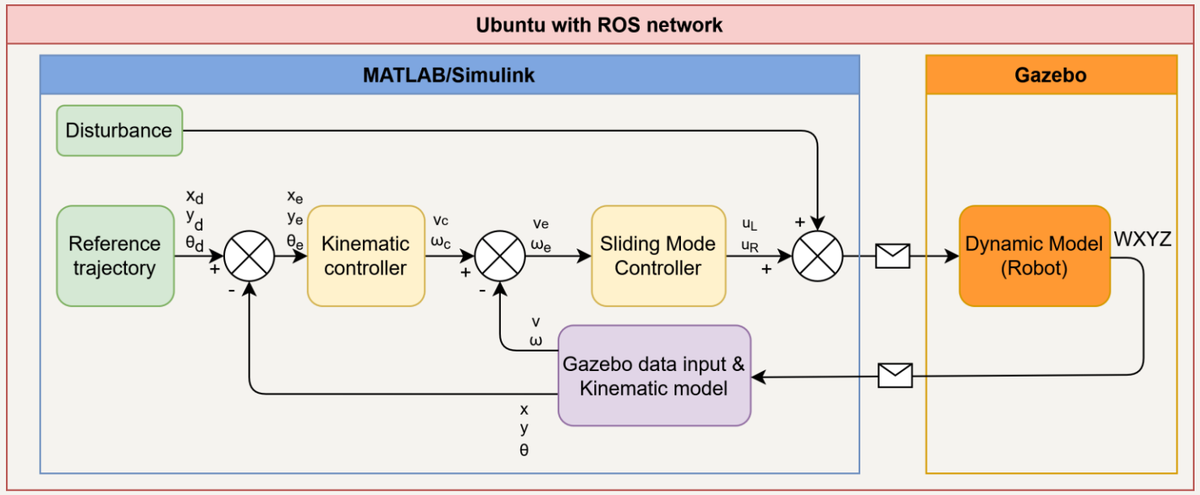
During my third year at university, I completed a large project over **eight months**. This was a **capstone project** which demonstrated my skills in:

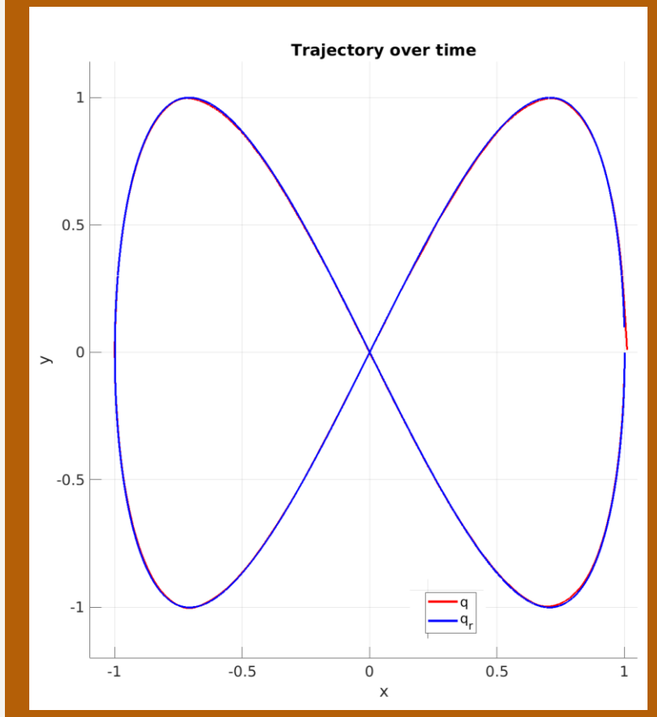
- **Research and Planning**
- **Project management and documentation**
- **Theory and implementation**
- **Simulation and evaluation**
- **Technical writing and presentation**

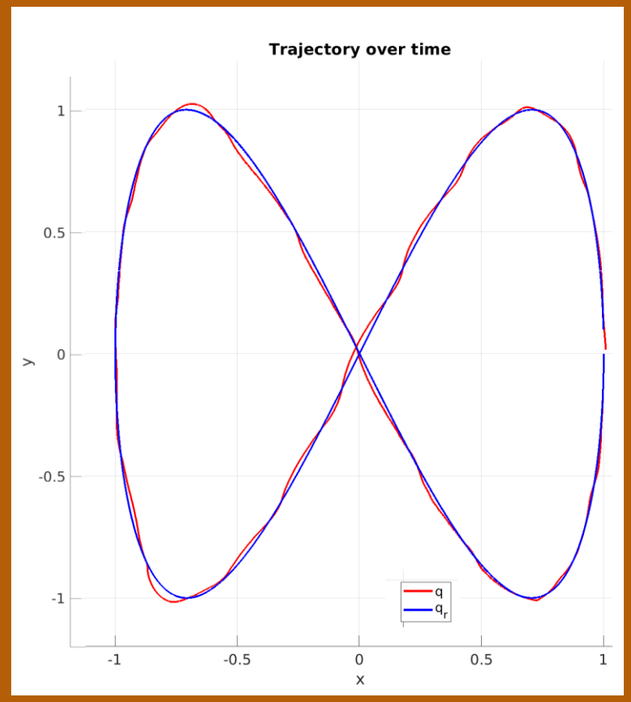
My project was to develop a **multi-agent controller** for a group of differential drive UGVs to **maintain a desired formation** whilst following a trajectory. The controller would ideally operate in a **distributed** manner and integrate with existing **Gazebo/ROS** systems from work by previous students. Whilst the main project was to simulate the controller, I also decided to do a practical element **and construct a third UGV** so that future students could continue work in the field.

The amount of topics covered by this project was extensive, and even an 87-page report at the end was not sufficient to explore everything. But a few are listed below:

- **Dynamic and kinematic controllers**
- **Differential drive motion**
- **State space and sliding mode control**
- **Multi-agent control**
- **Distributed cooperative control**
- **Graph theory**
- **MATLAB/Simulink**
- **ROS/Gazebo**
- **CAD modelling and design**
- **Manufacturing and assembly**
- **LaTeX document preparation**

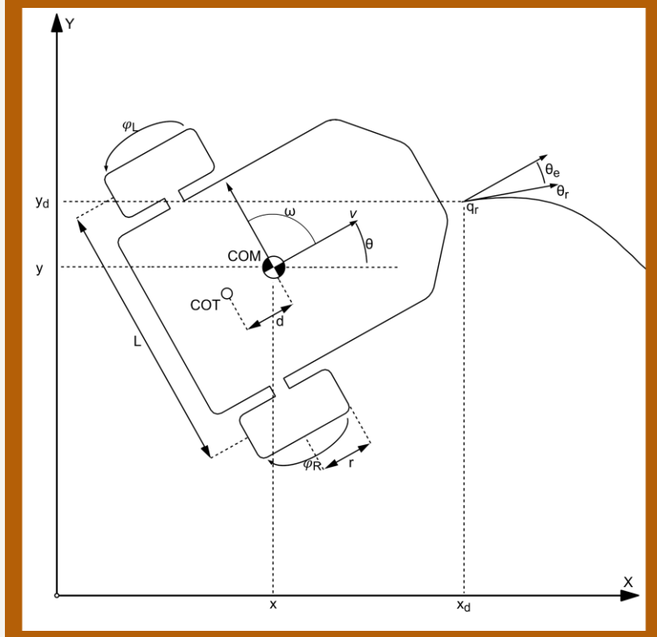


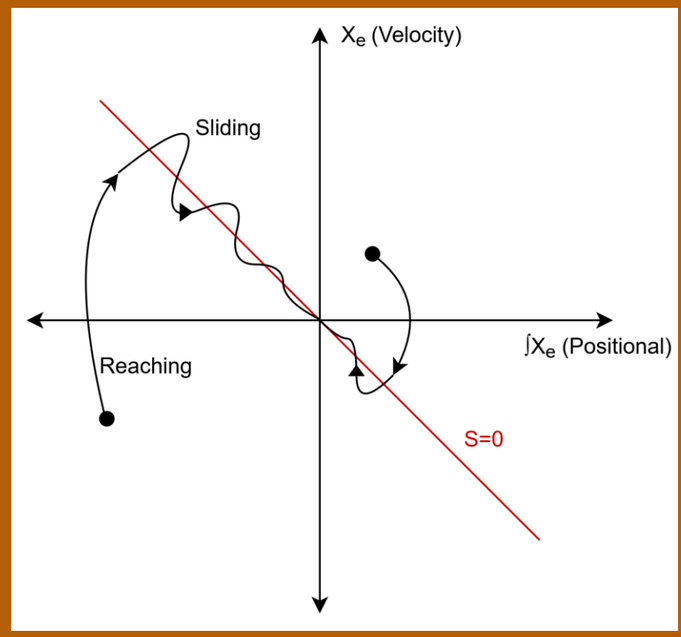




Above (left): UGV trajectory under ideal conditions using tuned sigmoid SMC, Above (right): UGV trajectory subject to disturbances.

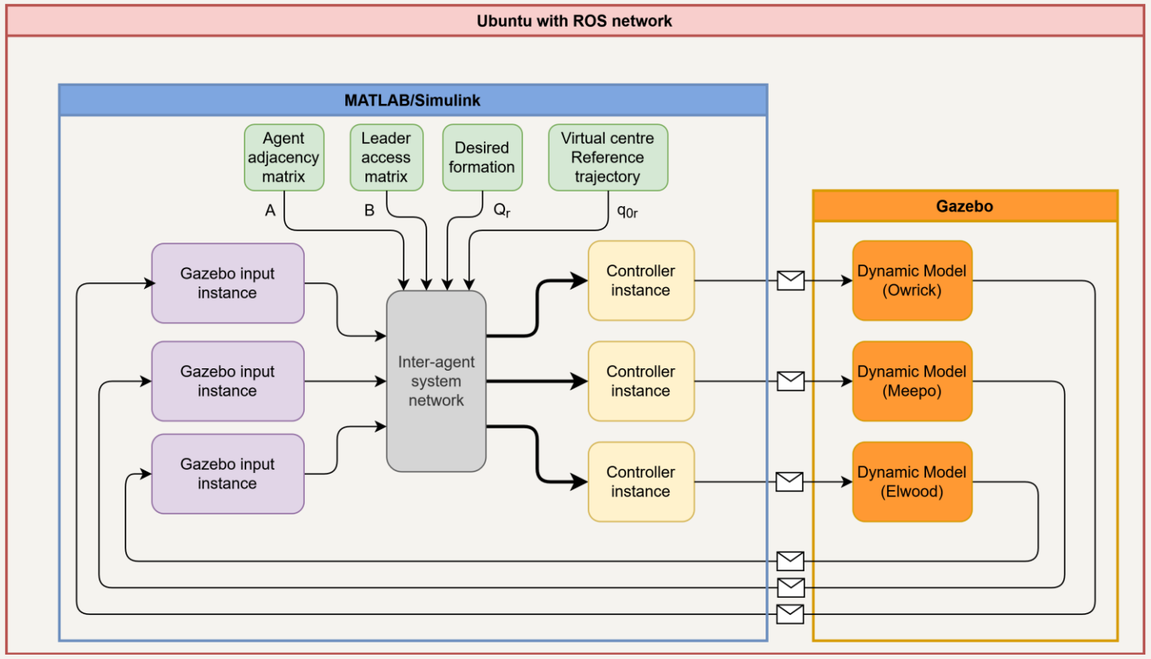
Below (left): Single agent graphic showing differential drive UGV and its physical parameters, its centre of mass (COM), its centre of turning (COT), current and desired trajectory etc. Below (right): State space diagram for UGV with illustrations of sliding surface minimising error with the reaching and sliding phases. The sigmoid function reduces chattering.





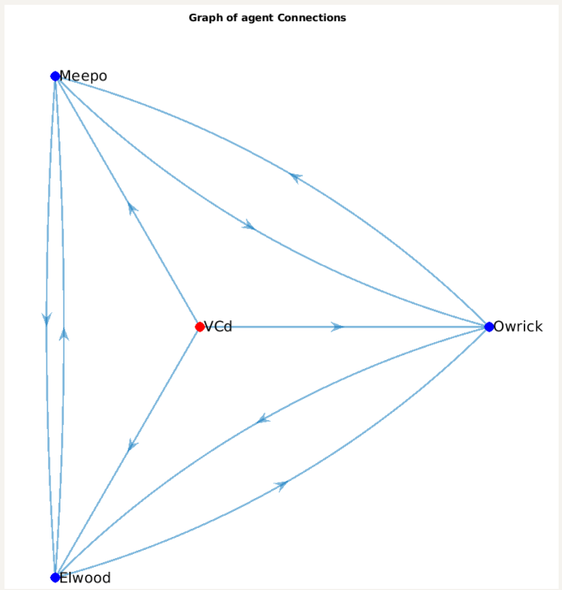
To the left: The system diagram for the single agent setup. The controller was implemented in MATLAB/Simulink and has a dual loop architecture, this means the controller comprises of a kinematic controller and a Sliding mode controller. Gazebo is a separate system used to simulate the dynamics of the UGV, information was passed between using ROS topics.

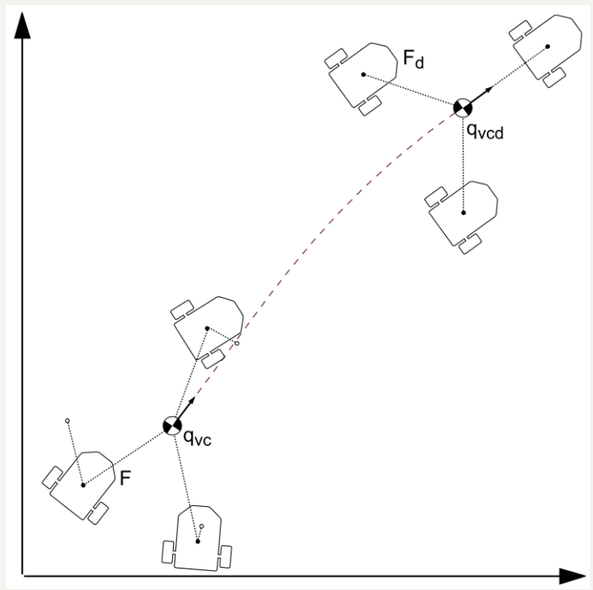
Multi-agent formation control of UGVs (part II)

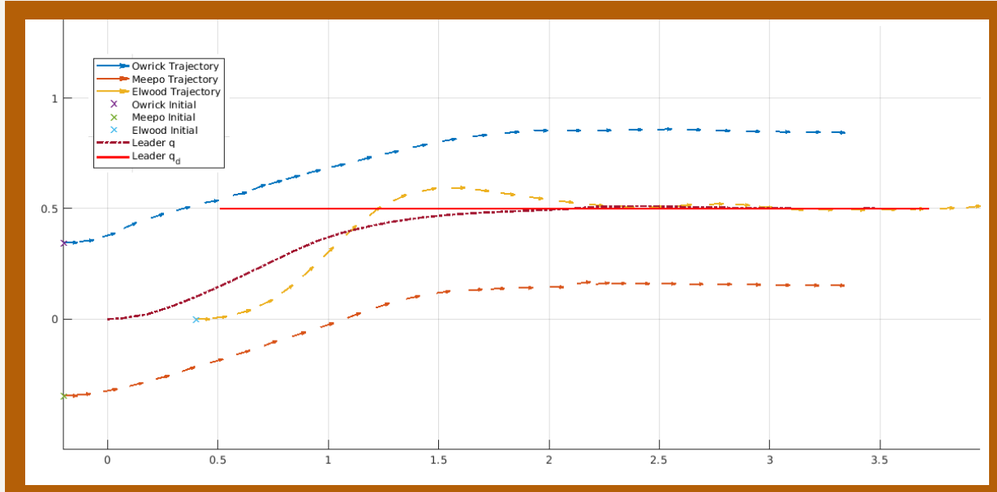


Above: System diagram for the multi-agent implementation. Note the additional robot instances, additional user defined parameters, and the **inter-agent network** which simulates the **communication between agents**. The network outputs **buses** for each agent containing information depending on the system topology (adjacent agent data, q_{vc}, q_{vcd}, F_d etc).

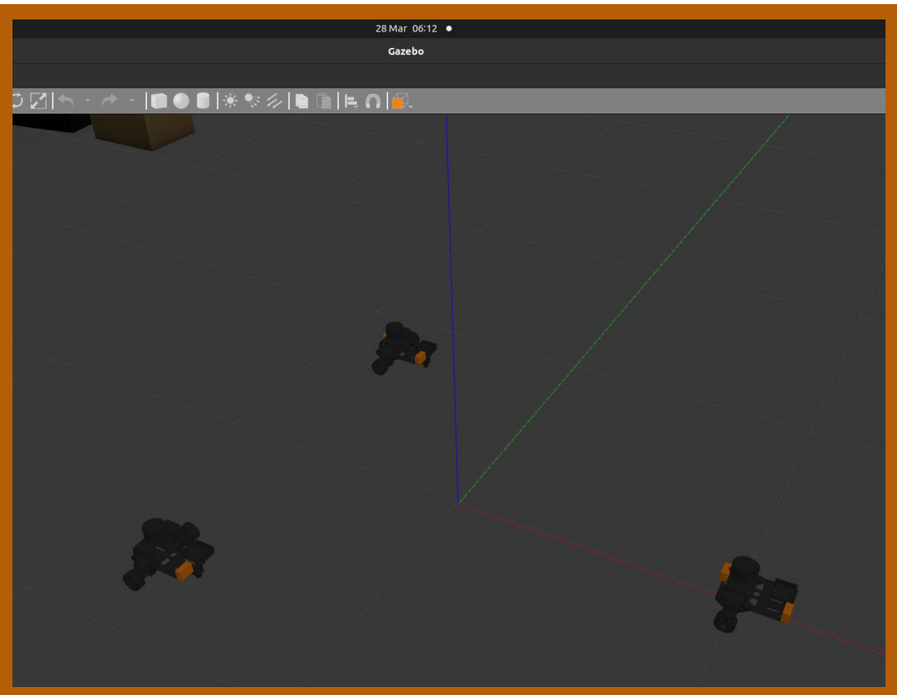
Below (left): system digraph showing a system topology with **ideal communication** between agents. **Below (right):** Multi agent graphic showing a current formation shape (F) and position (q_{vc}), and the desired formation shape (F_d) and position (q_{vcd}).



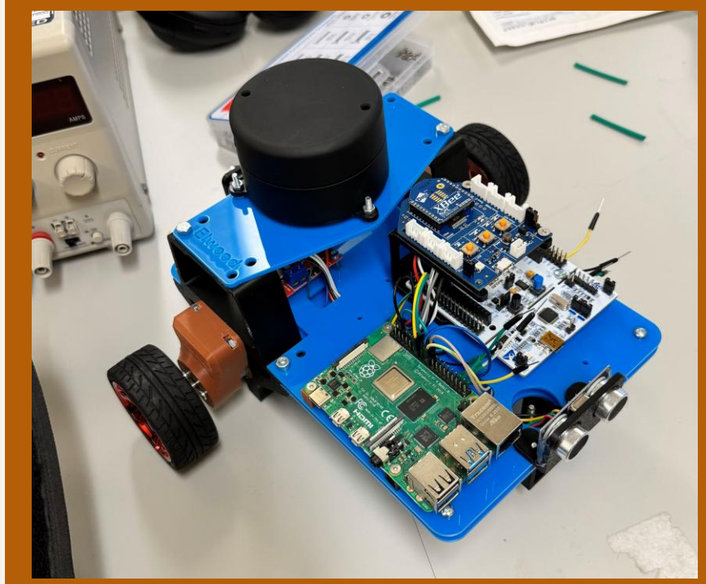




To the left: Trajectory of 3 agents and virtual centre reaching and maintaining a linear trajectory whilst keeping in an equilateral triangle formation.



To the left: Gazebo scene with the 3 instances of the simulated UGVs. This is the initial position of the robots. A startup script was used to set these.



To the left: My design for a differential drive robot. It's named "Elwood" and joins the two existing robots "Owrick" and " Meepo" from previous work. My new design aimed to improve the rigidity of the robot without significantly changing the overall design. This meant the previous robots can be upgraded whilst keeping the majority of the components.

PCB Art – Image2Kicad

Summer Project (May-June 2024)

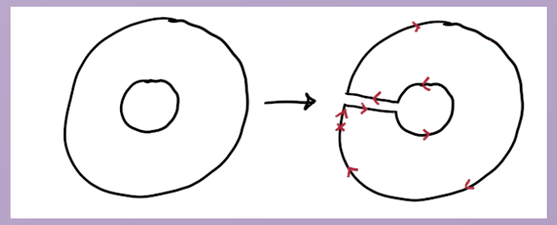
I wanted to put an image on a printed circuit board, rather, **make a representation of an image using the visible layers of a PCB to represent different colours.** I manually created masks with software tools like Blender and Inkscape, then aligned them all in KiCad. It worked but I wanted automate it.

So, I designed a Python program which converts **any** image into a native KiCad footprint, which you place onto your PCB design. It has a **simple, modern GUI** so that the user can easily choose an image, change mask thresholds and generate live previews of the final PCB, then they can press a single button to export to .KICAD_MOD!

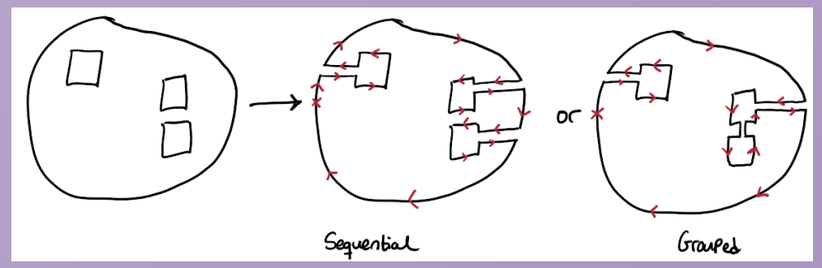
There are **3 steps** for the conversion.

1) **Mask generation** – where the image is converted into 3 high resolution **black and white** masks for the PCB layers.
2) **Shape generation –** The masks are converted into groups of contours, which are converted to KiCad shapes.
3) **Combination –** here the formatted layers are stacked together into one aligned footprint which can be exported.

Shape generation was the hardest to accomplish. This was because KiCad has **no way of representing negative space (holes),** but these "islands" cannot be ignored. I solved this by splitting the shape (shown below) and drawing the exterior and interior contours in one stroke.

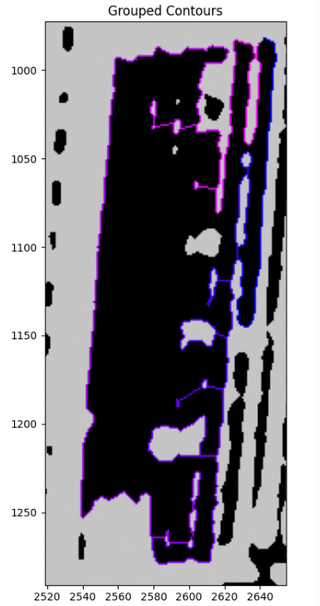


But what happens when there are **two islands**? Or **thousands of islands**? And you need to **connect them together** whilst avoiding crossing over others? So what is the best way to connect the islands? Finding solutions to these problems led to hundreds more that had to be solved.

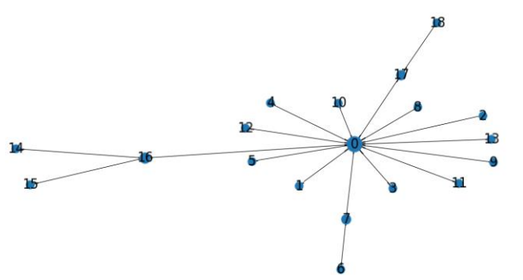


Working through these problems allowed me to learn many interesting algorithms and ideas. I discovered and applied **graph theory and minimum spanning trees (MSTs), NP** and **Travelling Salesman problems.** I built file conversion tools, many route-planning and minimising functions, and I debugged **visually** with plots and G-CODE viewers. I thoroughly enjoyed the process and I am very happy with the result.

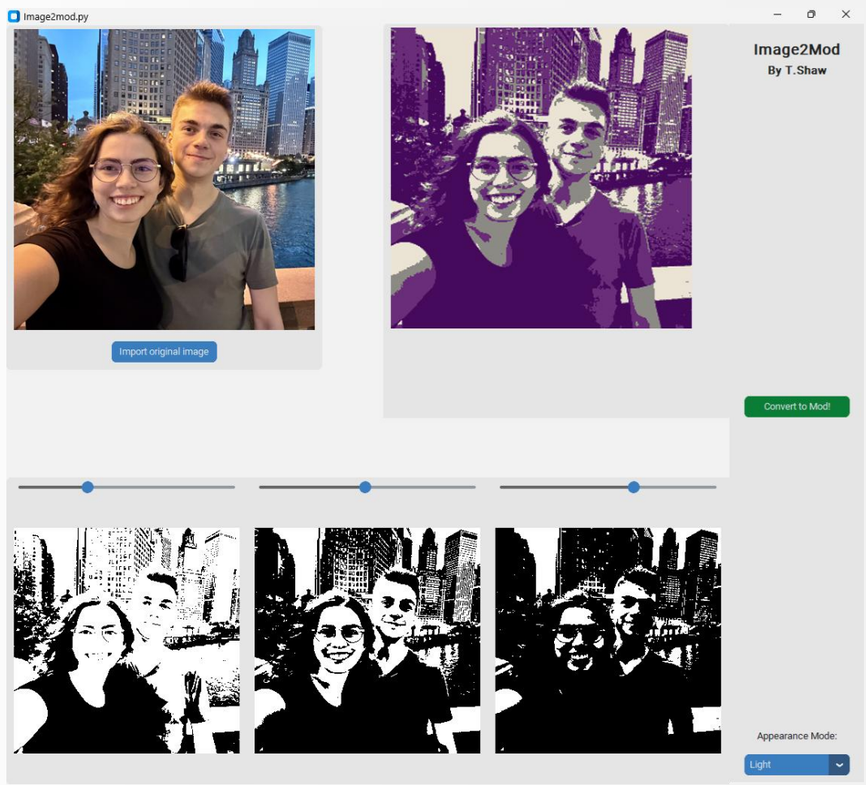
PCB artwork is a commonly added detail to custom PCBs by hobbyists, but they **often only scratch the surface** when it comes the available layers of a PCB (they use only the silkscreen). I used different layers of the PCB to represent different colours of the image. Together they can create a **more recognisable image** with **distinguishable features.**



Above: A troublesome region visualised with an early path tracer trying to reach all the islands in the **most efficient order**. Notice it misses some islands at the top but others are successfully traced.



Above: The minimum spanning tree (MST) for one of the regions' islands, notice the program has determined that island 14 and 15 can be linked to 16 for the best result.



Above: Screenshot of application interface I designed to simplify the process of creating the image art. Each threshold image and the final preview update live whilst moving the sliders. And it has a dark mode!

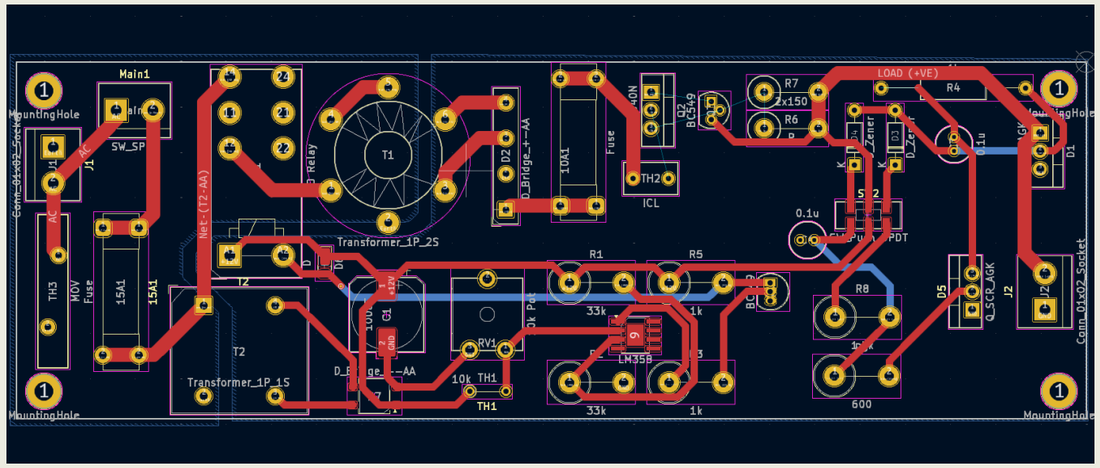
Below: a real demo PCB showing my image art.

PCB design practice

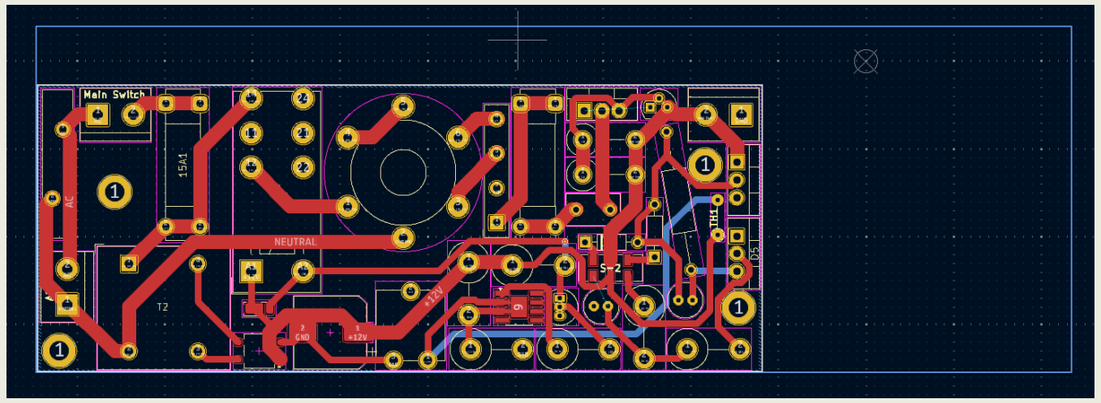
Summer Practice (September 2024)

For my first project, I created a 2-layer PCB for an AC to DC power supply I designed for my power electronics project last year. I had already found suitable parts and made a schematic as part of my project submission, all that was left was to assemble the components and route traces.

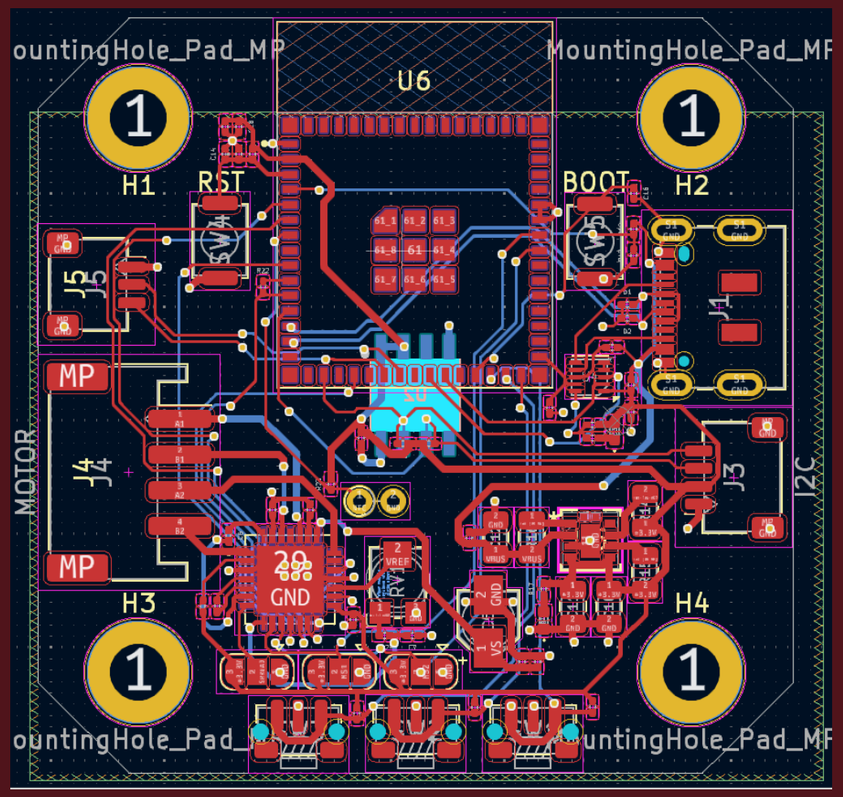
Since no footprint specifications were provided, I made a rough initial design that fit into a 150x50mm area which felt like an appropriate size. I made sure to use appropriate traces widths for the high and low voltage systems and I also physically separated them with a gap in the ground plane. This design is shown below.



From there, I improved the design by moving the high-wattage components to the edge to be attached to an external heatsink or casing. I also shrunk the footprint of the design by 42% whilst maintaining the isolation from before. This is shown in the below image



For my second project I wanted to practice designing 4-layer PCBs. I chose to replicate a Wi-Fi enabled stepper motor driver board I found online by Joshr120, where the schematic was freely available. I chose appropriate components and marked out the require shape to fit on the rear of a NEMA17 stepper motor. The PCB design is shown below.



The board had an ESP32 module with an antenna built in, so I made sure the copper ground fills did not interfere with this signal by completely avoiding that area. The motor driver was positioned away from other components and has room for a heatsink, and higher power traces to and from it are as thick as possible to reduce resistance.

Improvements: The differential pair for the USB had a troublesome route, and modifying the position of the USB-C port could improve the design. In addition, the mounting holes were initially made larger than the specified size by mistake, this led to a more cramped design than necessary and so the layout could be relaxed in a future revision. This would allow some of the power circuitry to be placed in a more optimal position.

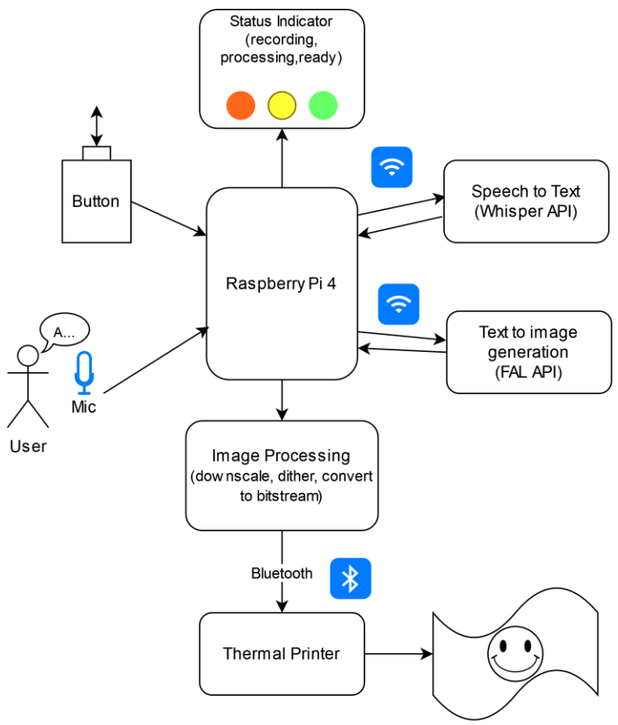
The Magic Microphone

2nd year Individual Project supervised by Prof. Tyler Bell (Jan-May 2024)

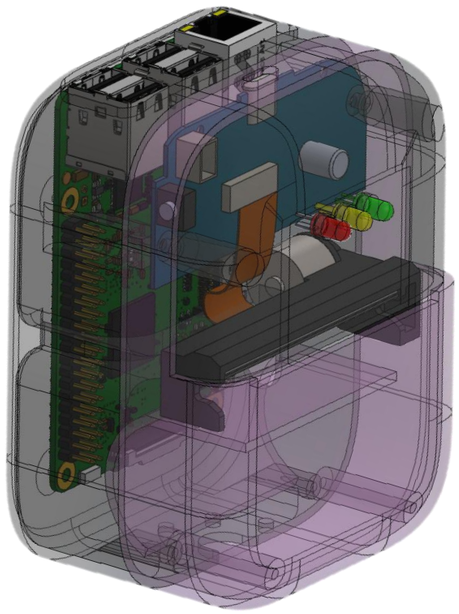
This project was intended to showcase the “Magic” of modern generative AI in a tactile and fun way. A handheld device which listens to the users' requests, and then prints out an AI generated image of their idea. It was inspired by a Polaroid camera, but in this case the **user simply explains their idea** and it creates an interpretation.

We wanted a **user friendly and approachable device**. The user should be able to easily pick up the device and immediately understand how to use it - simply hold record and speak. A red light means its recording, yellow means busy, and green means its ready.

Below (From left to right) : Mock-up of design, CAD model, final real product



To Left: System map
Below: CAD assembly of casing, raspberry pi, and printing hardware.



AI elements: Recorded audio from the user is sent to **OpenAI's Whisper** model, which transcribes the audio to text - we can run this locally but is much quicker to use their servers via an API (and remarkably cheap for the short audio clips we were using). The AI image generation was done using an API call to a service named fal, which provides a free text-to-image generator, so we took the text from whisper and combined it with a style.

“A brave knight defending its castle”

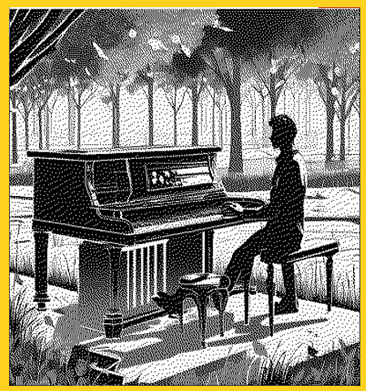
“A cat playing the piano, wearing a top hat”

‘A fierce pirate, comic-book style”

Printing: a Bluetooth thermal printer was used to create a 50mm wide black and white images on standard receipt paper. Using the python script the user can connect to the printer via Bluetooth. The images are then processed with a dithering algorithm which gives illusion of a larger color palette using different dot densities at the expense of resolution.

Greyscale Image

Without Dithering Algorithm



With Dithering algorithm

Whilst completing this project I exercised skills in:

- **Programming** on Raspberry Pi
- **Reverse engineering** existing devices (receipt printer)
- **Bluetooth and asynchronous** communication
- **Interfacing with APIs** and utilizing AI services
- **Prototyping** and 3D modelling

GAIT: Car short film

Dec 2023

For this project I wanted to try an unconventional approach to making AI generated video. I had experience in making short films with Blender, a 3D modelling program, so I decided to use AI tools to assist me. What would **normally take me months**, took **less than 2 weeks.**

First, I thought of a basic idea for a film, a car sat in a crowded, messy garage (like the opening scene of “Back to the Future”). Then I created a prompt for OpenAI’s ChatGPT **to take my idea and write a storyboard**. After some back and forth, I was happy with the story: a once glorious car, now sat in a lonely garage.

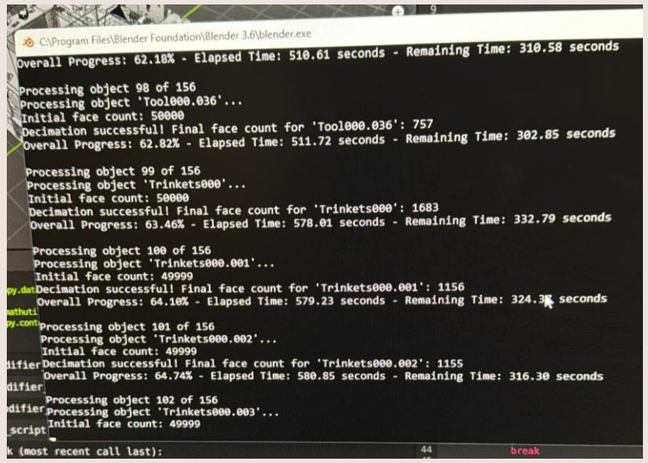
Then I asked for a **list of 200 3D assets** that would belong in that scene, I suggested categories such as:

- **Furniture**
- **Decoration**
- **Lighting**
- **Car parts**
- **Tools**
- **Desk items**
- **Cleaning supplies**
- **Trinkets**

For each item I used Luma AI’s Genie to generate a 3D model. These models were quite rough, but they are recognisable and came **fully textured.** I brought them into Blender and started scaling and positioning them to form a scene.

To the left:
Object view close-up of the Blender scene.

This project was for my “Contemporary topics in Engineering: GAIT” class,
You can see the final video here: https://www.youtube.com/watch?v=0rmKUwi74XM



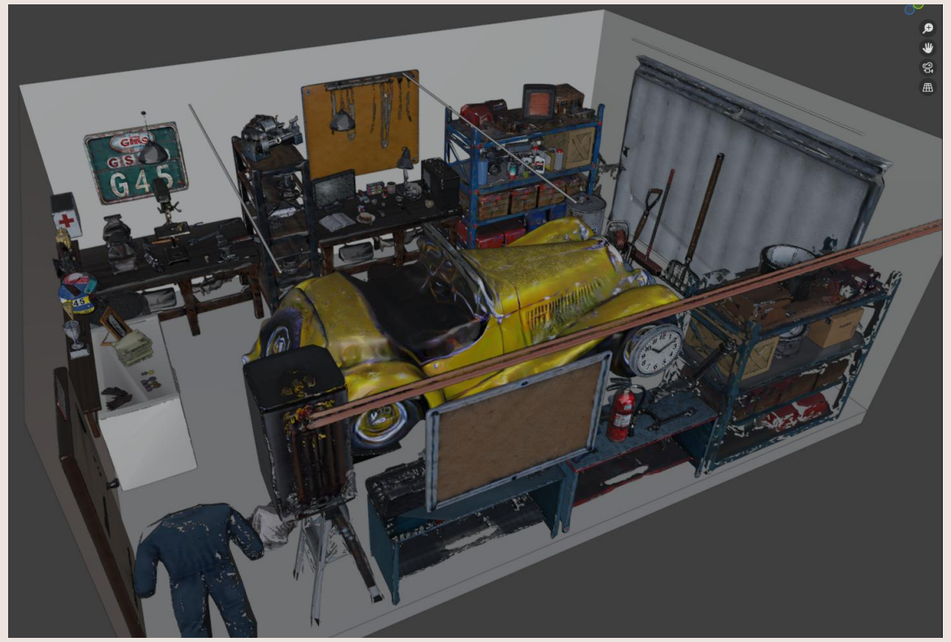
To the left:
Terminal-based updates from the decimation script used to reduce the complexity of the scene

After creating a basic camera sweep around the room, and the scene was done and it was time to render. Here I faced an issue where frames couldn’t finish rendering. **Each of the models had 50,000 faces** (regardless of its apparent complexity) which exceeded the memory of the computer. To fix this I worked with ChatGPT to write a script to progressively decimate the models, sacrificing quality to reduce time.

Finally, for the audio, the radio/commentator voice was generated by **ElevenLabs** (with script written by ChatGPT); and I used **Mubert** for the music and **StableAudio** for the sound effects (engine rumble and radio static), all were AI generated and really made the animation come together.

To the right:
The final scene in blender.

In total **185 AI generated models** were used.



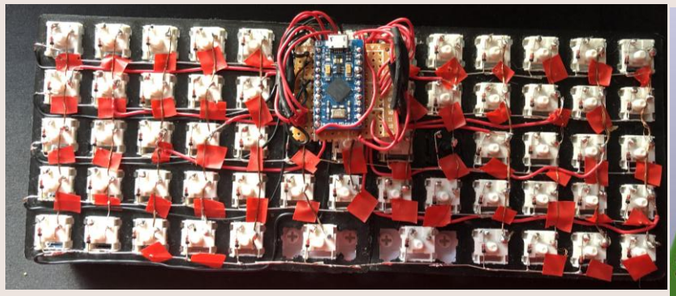
Custom Keyboard

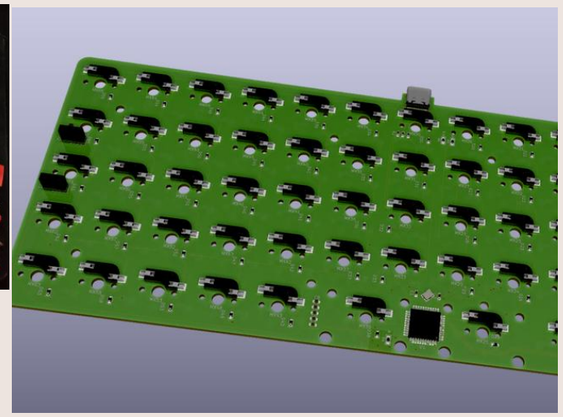
Feb-May 2021

In 2019 I built a keyboard from parts I found on the internet, a few years later I wanted to design more of the parts myself – specifically the casing and printed circuit board that was the centrepiece of this project.

For the PCB, at first I delved deep into the process of integrating a full microprocessor setup directly onto the PCB, this is a complex process and it was my first project using KiCad. I designed the entire thing and was ready to order but I found the cost of making prototype PCBs assembled with SMD parts was just too expensive for the risk that it wouldn't work. Disappointingly I could not justify the price.

So I redesigned the PCB to use an Arduino pro-micro which I tested beforehand with one I had already. This meant I could just order the PCBs and populate it myself with THT components. This was a much less complex design, effectively being a breakout board, so I was able to make the design more appealing.





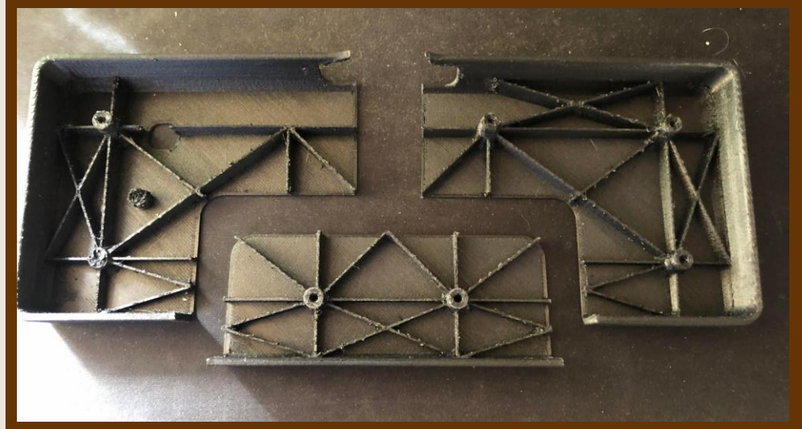
Above: prototype without PCB.
Right: Early PCB design with microprocessor
Below: final assembled PCB

For the case, I spent over a year making multiple prototypes before landing on a final design (v8). I had access to my Dads homemade 3D printer but it had very small bed so required splitting the model.

The v1 case comprised of 3 parts which I had to weld together with a heat gun, the case was flimsy and the USB jack was on the inside and was difficult to manuver whilst assembling. It was also quite uncomfortable to use, so improvements were clearly necessary.

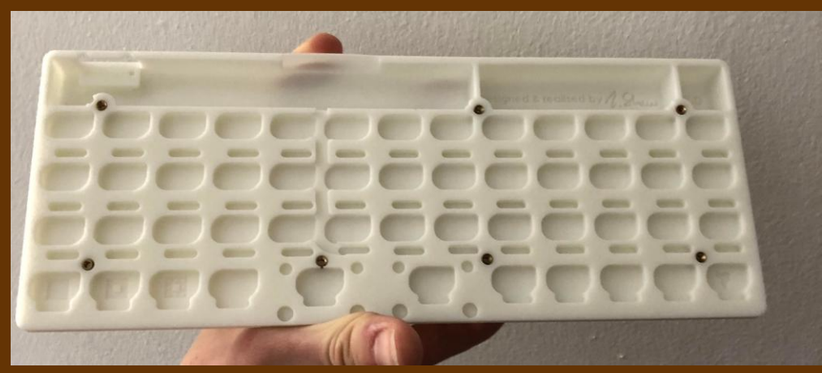
In cases v2-7 I solved the ergonomics issues by lowering the height and adjusting the angle. I added an USB-C daughterboard accessible from the outside to solve that. I also improved the structure to reduce flexibility, and ensured that the PCB was supported everywhere. I was happy with the design but I still had to join two pieces.

Finally, v8 was professionally manufactured with a resin printer. The improved manufacturing process allowed more detail and a smoother design to be made. I chamfered the edges, and added decals on the inside, my signature and a quote I loved. The part came back flawless and I am very happy with it. It made a great end to the overall project.



Above: v1 Case with 3 parts
Below: Right half of v5 case

Above: Close-up of decal on v8 case
Below: Full v8 case with threaded inserts



Custom Keyboard Video

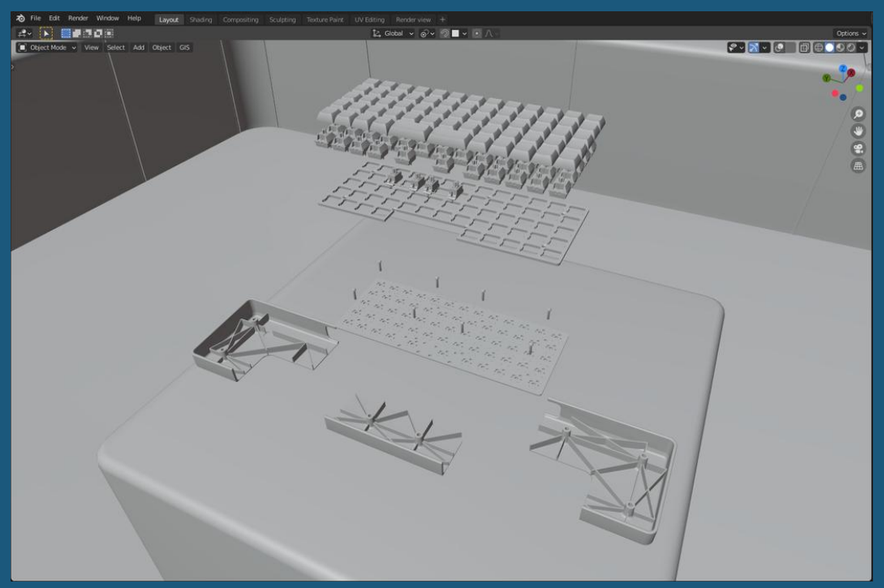
Feb-May 2021

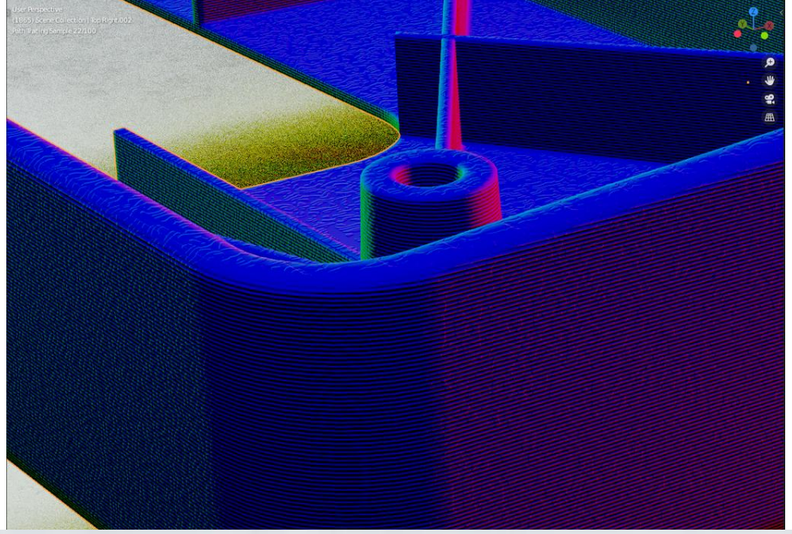
After designing the initial version of the custom keyboard, I wanted to create an animated short film showing it being assembled and shown off. I had recently become interested in using Blender and I was inspired by an advert from Nvidia for the DGX-A100. So, I challenged myself to make something similar.

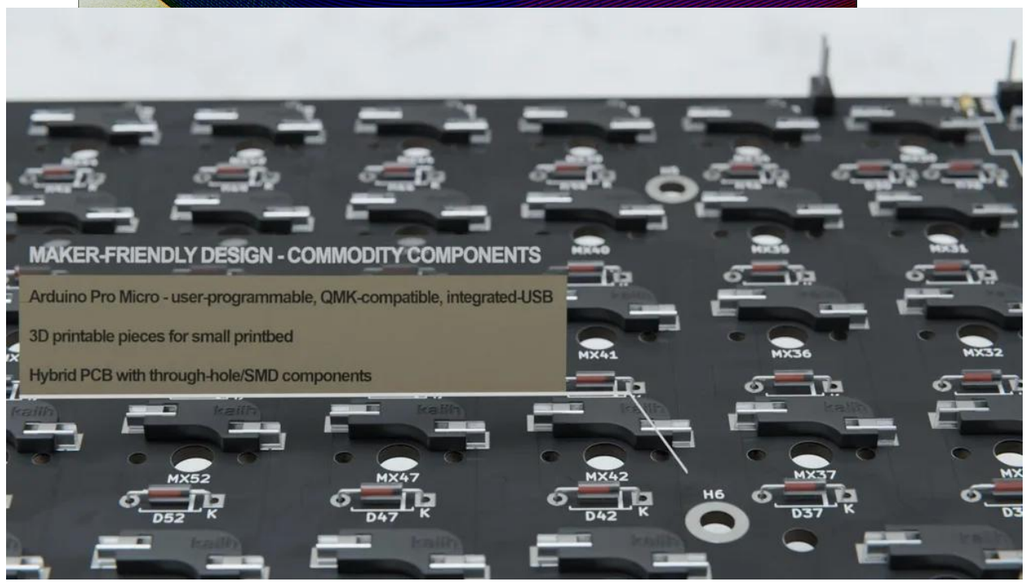
First, I made a storyboard, I decided the length of the animation as well as the key moments and camera angles. The scene is built around a room intended to look like the Deer Shelter at the Yorkshire Sculpture Park in the UK, as it fit the sci-fi, mysterious look I was going for.

Next, modelling. I had modelled my case in CAD and the common keyboard components could be downloaded, they could be quickly brought into Blender. The PCB was a major part of the design and has lots of important details which could be shown, so it needed to be accurate. KiCad can export 3D files but they don't come textured. So I exported individual layers as .pdfs and converted them .png with GIMP.

Texturing was crucial for the realistic look I was going for; I chose to create procedural textures for most of the scene as I was familiar with using nodes and I wanted to control the look of the parts myself. In retrospect, I should have used more image textures to speed up render times later.







Nodevember 2020

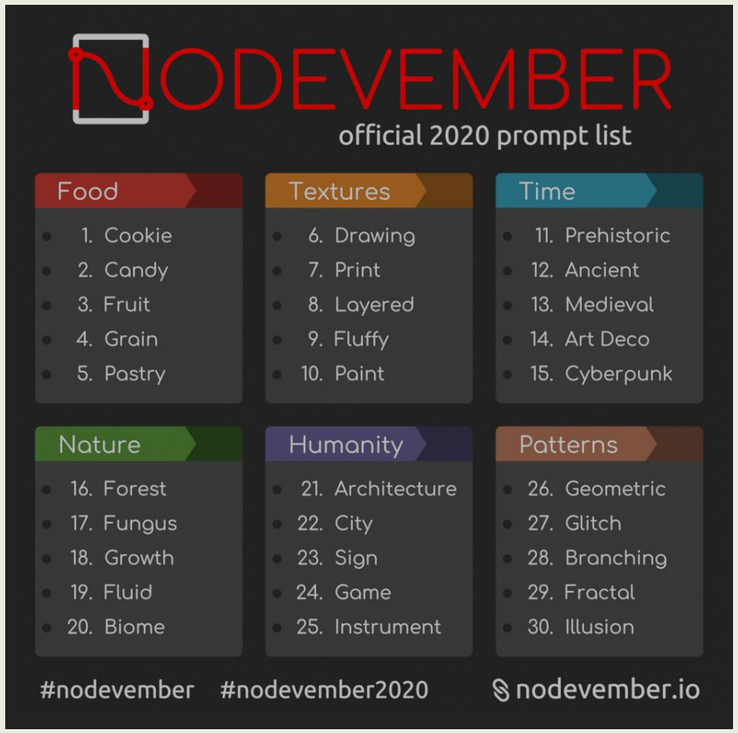
November 2020

During the first COVID-19 lockdown in 2020, I decided to learn the 3D modelling and creation suite Blender. I learned a lot in just a few months, and after a few months I had got comfortable with many of the design steps.

Nodevember is a yearly event in November for artists, each day a prompt is provided to challenge them to make something creative – but it must be made procedurally using a node-based programming language.

I took part in the 2020 event after watching a video on YouTube by @DefaultCube completing their first entry. I had some experience using Blender's material node to make textures but their video explained that a mathematical displacement map could be used to **reshape any primitive into any shape**!

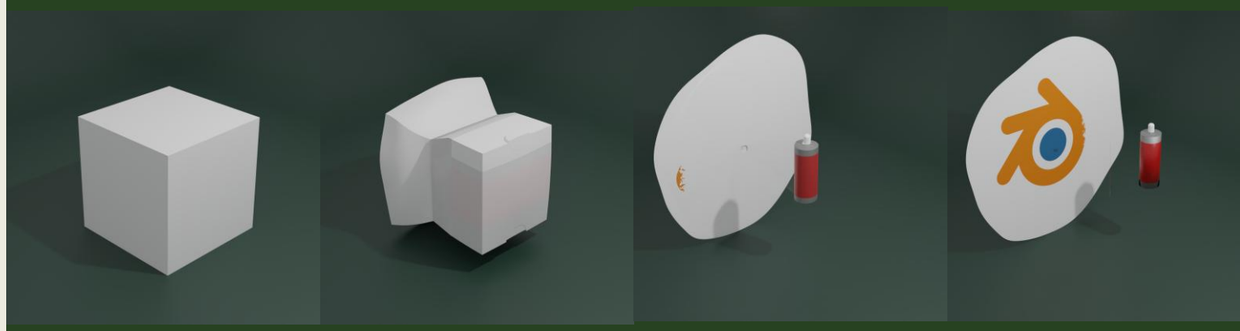
In this month I created 30 different Blender material node groups inspired by the prompts. Some were simply just textures, whilst others were full environments and 3D animated scenes – **all start off as a single cube!** On the left I show some of the highlights. Enjoy!



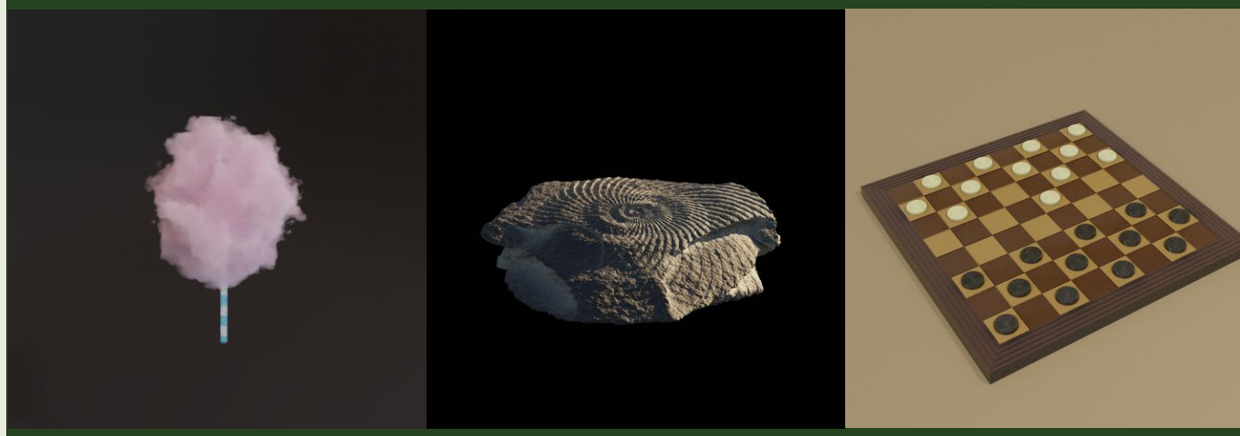
Full showcase video on YouTube: https://www.youtube.com/watch?v=Bu1SpBanWLs

Vector displacement is a method of displacing model geometry in more than 1 direction. As opposed to regular displacement (which displaces normal to the surface) it can shift vertexes in the direction of a vector specified by an RGB input.

Above (Sequence): Day 5 "Pastry" I wanted to make a waffle. This was by far my most expensive entry because my computer would crash when I rendered it and so I had to buy more RAM to be able to process the detail.



Above (Sequence): Day 10 "paint" was my favourite, an animation of spray painting a Blender logo, my only entry where the cube was split.



Above: Left - Day 9 "Fluffy", Middle - Day 11 "Prehistoric", Right – Day 24 "Game"