Catnip Craze Maze: A Co-op Game via Arduino and Unity

Technical Statement, CMPM Comprehensive Exam, Winter Quarter 2022
Samantha Conde - March 15, 2022

Summary:
In the Winter Quarter of 2021, I attended Game 202: Foundations for Alternative Controllers led by Professor Eddie Melcer at the University of California, Santa Cruz. This included (1) designing my own hardware controller from breadboarding to microcontroller programming, (2) utilizing serial communication to program event-driven scripts to interact with a cooperative multiplayer game, and (3) programming a game with procedural content generation (mazes) to interface with the hardware controller and complete objectives (collect catnip plants while running away from a dog). From this project, I got the chance to learn how to use an Arduino Uno microcontroller, code in the Arduino Editor using the C programming language, and work with hardware from wires to breadboards for the first time. I was and still am very new to coding and working with hardware, so this class took me out of my comfort zone and explored the possibilities of using an Arduino Uno, such as programming voice (volume) and analog buttons to influence the game environment. By the end of the quarter, I created and developed a working game called Catnip Craze Maze. Because this was a class project, Professor Melcer allowed the students to use any software compatible with Arduino code to run a game successfully. I chose to use the Unity 3D game engine as I had prior knowledge on the mechanics of the software.
Specific technical skills included: hardware controller prototyping, breadboarding, microcontroller programming in C, game programming in C#, serial data communication (Arduino to Unity), event-based scripting, and 3D asset integration.

The Game:
Catnip Craze Maze is a local cooperative game with up to 4 players, where the players help Marley the Cat get through a maze to get enough Catnip to satisfy him. The more Catnip he collects, the faster Marley moves throughout the maze. Marley is surrounded by 4 arrows shining different colors — green, red, blue, and yellow. Each player controls one arrow with the button corresponding to the color. Players need to communicate to decide when and where Marley should move. The location of the colored arrows can rotate at random times, so watch out for where your color will land next. Watch out though — there may be obstacles, like a Doggo, who will interfere with Marley's determination to get Catnip. So make sure you scare the doggo by meowing very loudly to get rid of him temporarily.

Technical Hardware Development:
I learned how to program an LED to turn on and off with a button click. Although this sounds very simple and easy for those who have been doing this for years, I was surprised when I finally got it right. There were a few hiccups as it did not work the first time I put it together. It turned

out I had misplaced the ground and power wires. After switching the two wires, the light turned on and off with a button click.

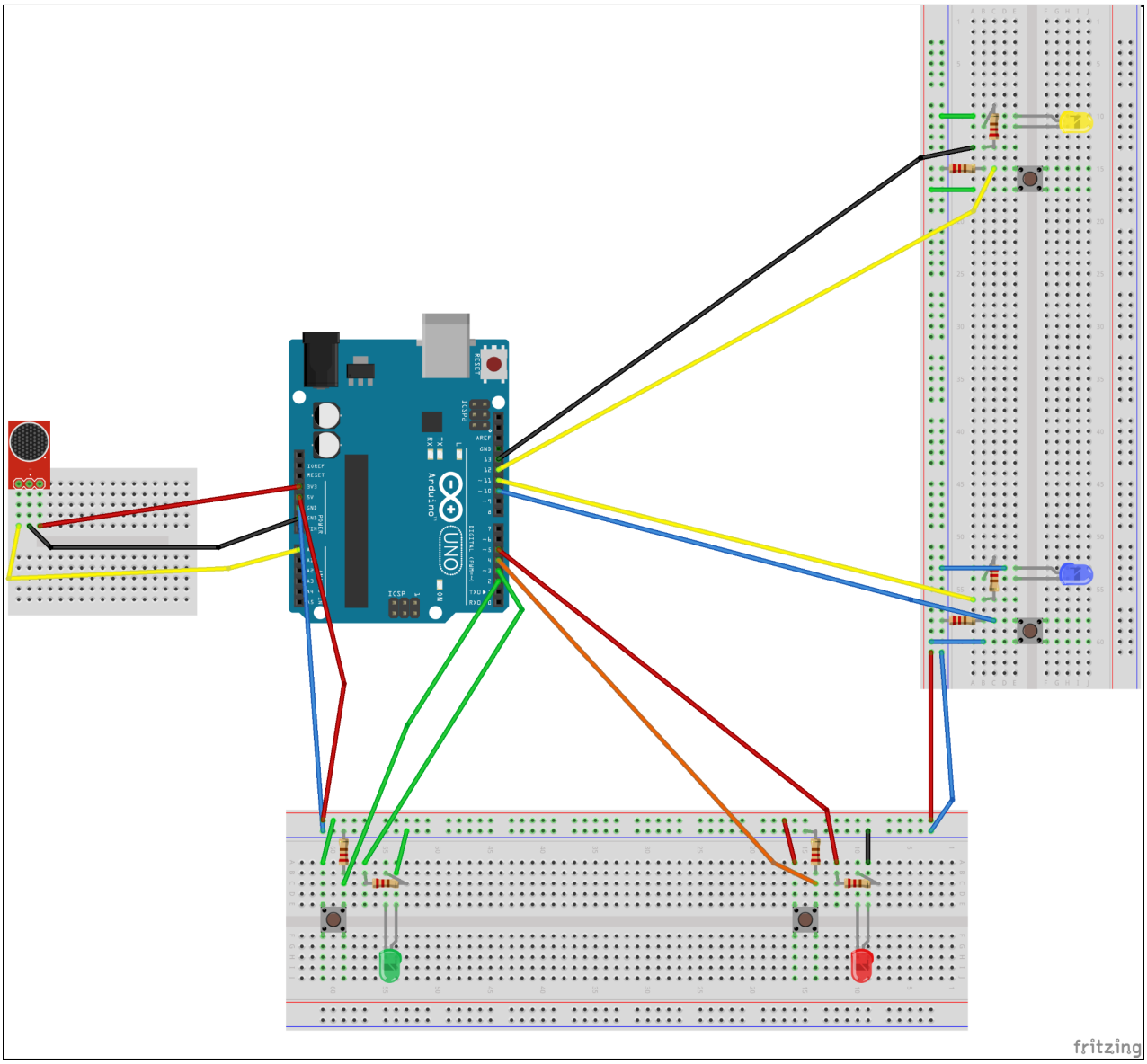

Figure 1: A Fritzing Image of the Arduino Uno and the Breadboards

We also learned how to program and use the microphone that came with the Arduino kit in class. After writing the code for the microphone and running it, the Arduino Editor was saying there was an error. I began reading line by line and was stuck not figuring out what was wrong. So I asked someone to advise me on what I was doing wrong. It turned out I missed a few semicolons. Syntax errors are not fun but can make your code not work for the smallest mistakes.

I took the two codes for the LED and microphone and created a new script meant for the game's interface. At the end of designing the idea for Catnip Craze Maze, I had a system consisting of 2 big breadboards and 1 mini breadboard with 4 buttons, 4 LEDs, and 1 microphone. I had to make sure the code included all the LEDs, buttons, and microphone corresponding to their pin locations on the Arduino Uno [Figure 1]:

```
int micInputPin = 0;
int buttonGreen = 2;
int buttonRed = 4;
int buttonBlue = 10;
int buttonYellow = 12;
int LEDgreen = 3;
int LEDred = 5;
int LEDblue = 11;
int LEDyellow = 13;
```

Game Programming:
After writing the code for the hardware, it was time to switch to designing and programming the game in the Unity game engine. For starters, I had to create a plane to hold a maze that the Cat game object (Marley) would move around in. By learning online, I downloaded, designed, and adapted a free maze generator script from the Unity Asset Store. I adapted the code to generate a new maze every time a new game started.

The next task was to place Marley the cat, the arrows, the Catnip, and the dog (Doggo) on the maze. As I did not have the skills to make my own models, I searched online for game objects to use such as a Cat.obj, Dog.obj, Catnip.obj, Arrow.obj, and a Hotdog.obj. I downloaded the game objects and modified their size and color patterns in Unity.

The first code to write was to make sure Unity could read the script from Arduino. I collaborated with Joseph Adamson and Aviv Elor to write a script to connect the microcontroller from Arduino to Unity.

The next code to write was for Marley the cat and the arrows surrounding it. The first task was to make sure Marley was placed on the Maze and moved around. Through Unity's Scripting API documentation and Unity Learn Library, I taught myself to write the code to move Marley around the maze with arrows. One significant error was that the Marley game object would fall completely through the map and go through maze walls. I needed to set up the collider physics within Unity, so it was a simple fix. The next thing programmed was to rotate the arrow's location after a random amount of time; this idea was inspired by *Octodad: Dadliest Catch*.
All of this was part of SerialReader.cs

After the Marley gameobject was ready to go and was able to be moved by a click of a button, I moved on to writing the code for the Catnip gameobject that the Marley gameobject will come in contact with and collect. I learned how to develop custom prefabs to streamline game assets in Unity which made coding the Catnip gameobject easier. I adapted the random maze generator

plugin to ensure the Catnip gameobjects appeared randomly throughout the maze. I added a fun little script where the Catnip gameobject would be spinning around in place (this was on a separate script called Rotator.cs). The last thing to code, when the Marley gameobject would come into contact with the Catnip gameobject four things would happen: 1.) A short purring sound effect would play 2.) the Catnip would disappear 3.) it would add to the`KittyCrazeMeter` that would let the player know how many Catnip they have collected and make Marley move faster throughout the maze (making it harder for players to control Marley).
All of this except for the rotating script was part of SerialReader.cs

To make the game more interesting, I decided to add an enemy also known as Doggo. The Doggo gameobject was coded to start at the opposite side of the map and track Marley gameobject so they would follow and come into contact with it. When the Doggo gameobject was closer to the Marley gameobject, a short sound clip of a dog barking played to warn the player the Doggo was near. If the Doggo gameobject touched the Marley gameobject, Marley would flip on it's side and the game would end. To stall Doggo, the player would have to MEOW into the microphone and the Doggo gameobject would exchange for the Hotdog gameobject, a short clip of a dog whining would play, and the gameobject would freeze for a few seconds. Mostly all of the Doggo code was in Doggo.cs except for flipping Marley – that was part of SerialReader.cs

I was not too fond of the final look because it was missing a mini-map. So I learned how to code and control the camera settings in Unity for a mini-map.

If I were to bring this project back to life, I would integrate a tilt sensor/switch to the hardware. The tilt switch could be programmed to a new enemy element of a human hand coming to pet Marley. The player would have to hit the tilt switch to stun the hand in the game.

Related Links:
- Indiecade Horizon 2021 Game Demo Video: https://www.youtube.com/watch?v=VM8NJARjiug&t=549s&ab_channel=UCSCHorizons
- Github Repository and Game Code: https://github.com/samconde/CatnipCrazeMaze

Individual Contributions:
- Hardware prototyping and development of the custom game controller, including the microphone (voice), analog buttons (press), and LED (indicators)
- Game development from applying procedural content generation maze, 3D assets, and event-driven programming

Acknowledgements:
- Aviv Elor: advising on resources for Unity Game Engine Programming and Arduino Serial Communication
- Joe Adamson: advising on breadboarding and hardware debugging.
- Pavlo Vlastos and Conrad Esch: playtesters for Catnip Craze Maze
- Professor Edward Melcer: advising on project ideas and sharing iterative feedback