The Jewels of Citrine City

**A Physical Puzzle Game developed by George Linfield**



**Introduction:**

For days you have been wandering the desert. Your supplies are running low, and the excitement of discovery and adventure can only fuel you for so long. Then, just as hope begins to fade, you catch sight of an orange glow in the distance.

It can’t be. It must be. The legendary Citrine City, long thought lost to the dunes of the desert. You approach, hoping for nothing more than a meal and a rest.

But the amber glow of this once great city, now dilapidated and empty, masks a secret. One last treasure that the city still hides. Can you solve the puzzles and unlock the secret of Citrine City?



**Concept:**

Inspired by 80s and 90s game shows, *The Jewels of Citrine City* is a riddle-based puzzle game that takes place in a 3D tabletop space. You are a traveller who has stumbled upon Citrine City, a once majestic town carved out of orange brick and gemstone, now reduced to a dilapidated and empty ruin. There is a hidden treasure in the city, and only by solving the puzzles set out by Marvellous, the mysterious guardian of the city, will you be able to take it.



**Development Documentation:**

**| Initial Design**

Before the concept for the game came about I was actually working on an interactive music box that would respond to player proximity by playing music, and then respond to the player singing along to the music by making a diorama move. My long-term vision for this was to utilise it in an immersive theatre piece – it could be hidden somewhere in a room, activate when a player approached, and then reveal some hidden details about the world of the play.

With this thought in the back of my mind, I decided to change the moving diorama to a set of gates that could conceal hidden objects (with the purpose of providing additional exposition to the world of the play). Instead of playing music, the music box would now deliver a clue, and the player would have to respond with an answer to open the gates.

**1** Rough Schematic of Initial Design

Technologically, the sound recognition that this call and response would require was beyond my physical computing knowledge, but I was struck by the idea of incorporating puzzles into the interactive music box (which at this point wasn’t really a music box, as it no longer played music nor responded to singing…). I devised an alternative to the sound recognition – a hidden microphone that the player would have to find in the room and speak into to open the gate – and the puzzle game was beginning to take shape.

****| First Prototype**

**2** Wave Shield mid-soldering

It was at this stage I started putting together a prototype of this simple puzzle. I had already decided to use an Adafruit Wave Shield V1.1 to add high quality audio playback functionality to the Arduino. Putting this together was a challenge in itself – there were about 200 joints to solder on the shield, including additional header pins to enable continued access to the Arduino to allow me to hook up additional sensors whilst using the shield, and I hadn’t done any soldering before. However, Adafruit provided some great instructions along with the kit, and putting it all together only took a couple of hours.

The shield came with two sets of code to simplify accessed the .wav files on the SD card. Both used the WaveHC library. I initially tried dap\_hc, but then swapped to play6\_hc as it was easier to specify which file you wanted to play at any one time, and also made it easier to call these files within the code.

Whilst it was easy to assemble, when I first tested the shield it didn’t output any audio. When troubleshooting using the Serial Monitor I discovered that the shield was reading the SD card fine and was successfully playing the audio (I printed continuous dots to the Serial Monitor during playback). This meant that the problem was with the headphone jack or the volume potentiometer. I checked my solder joints and re-soldered a couple of the potentiometer joints (I initially didn’t use enough solder) yet still the problem persisted. It was only after removing the shield from my Arduino to check the soldering again that I worked out what the problem was. The metal casing of the power socket on the Arduino was touching the potentiometer joints when the shield was slotted onto the Arduino, creating problems with the circuit. I clipped the joints as short as they would go, and tested again – the shield output perfect audio. Success!

I was now ready to add the two ultrasonic sensors that would be used to detect when a player approached. Unfortunately, I could only ever get one working at any one time, and after spending a fair bit of time troubleshooting this I decided that it wasn’t worth the time and effort to have two ultrasonic sensors (which was initially desired to improve the precision of the player detection) when a single one would still work fine.

Finally, I added the microphone. This was where a third problem arose. I had ordered a MAX 4466 microphone, and assumed that this would work straight out of the box. However I didn’t realise that I would need an additional amplifier to get the level of readings that I needed for this project – I should have done more research before ordering the parts. Without an amplifier, the microphone wasn’t strong enough to pick up sound without being literally right next to the sound source, and even then it was still weak. Unfortunately at this point in proceedings it was too late to order an amplifier, meaning a crucial part of my initial design wasn’t going to work.

**| Iterated Design**

The problem with the microphone, and the limited time I had to fix this, meant that I had to return to the drawing board. I was pleased that the proximity sensor could be used to trigger the clues, but I now didn’t have anything that the player could use to respond to it. Whilst testing my microphone I did discover that when doing a peak-to-peak reading it would rapidly spike when pressing a finger up against the microphone. I decided I would incorporate this into my iterated design and use it as a touch sensor that would start the project instead of the ultrasonic sensor. This freed up the ultrasonic sensor to be used in one of the puzzles.

I still wanted to use something instead of the microphone, and settled on a simple light resistor that would be encased in a lightbulb, as this idea fitted in with a riddle that I was planning on using as one of the puzzle clues.

It was at this point that I began to think seriously about the backstory that this game could have. The game now had three puzzles – based on a ‘touch’ sensor, an ultrasonic sensor, and a light resistor. I wanted to create a world that could provide a backdrop to these puzzles, and a consistent theme to the puzzles themselves. For the latter, I opted for riddles (mainly because I’d already written one…). I also really liked the idea of a benevolent taskmaster who would offer clues to the player as they went (harking back to *Crystal Maze*, a big inspiration).

**3** The Crystal Maze

I took some time away from the electronics to write out the story of Citrine City, a once bustling city now lost to the sands of time. The city was carved out of orange rock and gemstone in a desert, and used to be full of keen puzzle-solving denizens. They have now all mysteriously vanished, and only the city’s guardian, Marvellous, remains.

Marvellous would be the character that would deliver audio clues to the player, and I decided to try and build a small model of Citrine City that would house the physical puzzles. The model would also have big servo-operated gates that conceal the city’s treasure, ready for when the player completes all the puzzles.

At this point in the design process I had a much clearer idea of what I wanted the final product to look like, so I set to work on putting all the pieces together.

**| Second Prototype**

I already had the Wave Shield to play music, and I quickly hooked up the repurposed ‘touch sensor’ microphone and the ultrasonic sensor. I also added a light resistor that I placed in a lightbulb, to use as the final object clue. The values these output would be used as triggers for the puzzles, and when a value changed they would advance the game state. Finally, I added in two servos to operate the gates.

I used Boolean variables to indicate whether a riddle was solved or unsolved, and a second set of Boolean variables to keep track of the game states. When a riddle was solved, the servo would move and an audio clip would play.

The second prototype went far smoother than the first. One of the massive problems I ran into with the second prototype was getting each component working in tandem. I initially checked that each sensor was working correctly in isolation, and then copied the code over to the relevant section of the game system. However it often required a fair bit of tweaking to get the code to play fairly with the Wave Shield libraries. I had to swap to a new Servo library (ServoTimer2) as there was a conflict between the included Arduino servo library and the Wave Shield.

**4** Printing components

Once all the circuits were working and operating as intended, I began to create my game board. To fit my theme of an orange city I 3D-printed as much as I could in orange. Unfortunately due to the length of time it takes to print, I didn’t have time to make a case for the Arduino, or a base/backdrop. Instead I used cardboard, and decorated it with orange vinyl. Finally, my course-mate Nick, a talented artist, drew insignia for the doors.



****

**5** The finished game

**| Further Developments**

Due to time constraints, I had a limited amount of time to playtest the game before presenting it, and this meant there were a few things I would change about the play experience.

After play-testing it became clear that I would need to add more interactive objects to act as ‘red herrings’, as in its current limited state the game was far too easy. I also want to add further puzzles. I think the idea of using a riddle as a clue to which object to interact with worked really well, and there is definite room to take this idea further.

I also want to build a proper cabinet to enclose everything, as the makeshift cardboard isn’t the best look. I may look into using shelves to hold the different objects rather than laying them out on a canvas.

**What Went Well?**

* I developed several new skills. Before working on this project I had very limited knowledge of physical computing. I’d never done any soldering before, and didn’t know how to design 3D objects for 3D print. I quickly picked up these skills, and also improved my ability to troubleshoot hardware. I learnt a bit about different Arduino libraries as well.
* I played to my strengths. Given the gaps in knowledge listed above, I think my initial design concept was a bit beyond me, so I went back to the drawing board and brought in more creative writing and world-building, something I have great experience in. I also leant on the code to create a play experience, using the sensors as control inputs. Whilst this meant the end project wasn’t as perhaps exciting as it perhaps could be, it did mean I had a more solid final product.



**What Could Be Improved?**

* Conceptualisation and planning. I ended up stumbling upon my final concept because instead of thinking of multiple initial ideas I became fixated on one idea that I wanted to try and make work. I think this is in part down to the way that I work – constantly iterating. However this meant that I ended up putting the pieces of my final project together very late in the day, instead of having a strong basis of what I wanted to do that I could then follow.
* Troubleshooting. I lacked the knowledge to quickly identify problems and this meant I lost a lot of development time chasing shadows. The end result had one or two bugs that I unfortunately couldn’t iron out for the final showcase as a result.



**Appendix**

**| Libraries used**

For the servos: [ServoTimer2](https://github.com/nabontra/ServoTimer2/blob/master/README.md)

For the Wave Shield: [WaveHC](https://github.com/adafruit/WaveHC)

**| Tutorials**

Wave Shield: <https://learn.adafruit.com/adafruit-wave-shield-audio-shield-for-arduino?view=all#wavehc-library>; https://learn.adafruit.com/adafruit-wave-shield-audio-shield-for-arduino/play6-hc; https://learn.adafruit.com/adafruit-wave-shield-audio-shield-for-arduino/convert-files?view=all#faq-7

Operating two servos with Arduino: <http://www.robotoid.com/appnotes/arduino-operating-two-servos.html>

Adafruit Microphone MAX4466: <https://learn.adafruit.com/adafruit-microphone-amplifier-breakout/measuring-sound-levels>; <http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-sound-detection-sensor-tutorial-and-user-manual/>

Plus the Arduino Reference database.

**| Schematic**

