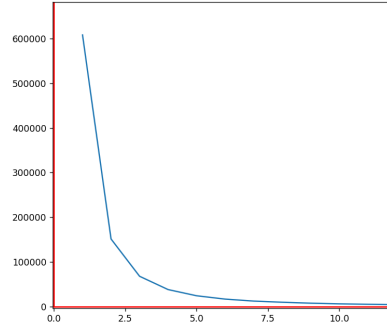


This Logbook will be updated weekly. Most of the content will compile the contents of [this changelog](#).

Date	Work Completed	References / Links
2021/12/20 to 2021/12/26	<p>I began to work on my project in earnest. Due to previous miscellaneous prototyping, I had some basic game assets and a rough idea of how to use Pygame. Initially, I was confused and unsure of how to begin the game itself, since I don't have a clear direction in which I want to go with my game. I solved this by deciding to instead focus on the tools I would need to make the game.</p> <p>I first created a repository on Github, but I was tired so I just left it at that for the night. The next day, I added all the concept files that I had leftover from throwaway prototypes, a collection of generic Pygame functions and the rough game assets. I labelled this phase the Pre-Alpha phase since I had not begun to do any work yet.</p> <p>I had my first major breakthrough when experimenting with shortcut (.lnk) files, creating a function that can read the target directory of a shortcut file. By opening the shortcut file in read binary ('rb') mode, I obtained a list of ASCII codes. By linearly scanning for the beginning and end of the target string, I could extract this target and append it to the name of an asset file in order to produce the asset's entire path. Therefore, I was able to have GameAssets and Scripts on the same directory level, rather than having GameAssets within Scripts.</p> <p>I initially had great difficulty with implementing this, since searching up "access shortcut file target in python" did not yield any helpful or comprehensible results. I found my first clue when I saw that a user on the website <a href="#">Stack Overflow</a> used <code>open(path, "rb")</code> as the first line of their code solution. From this, I did extensive experimentation, which ultimately led me to the solution. This tool will greatly aid me in the future development of my game.</p> <p>My second major breakthrough, and the first directly pertaining to the game's code design, was a level file loading system. The system can read a file written as a python list of <code>Element()</code> objects, and then load every <code>Element()</code> onto the Pygame screen. As understanding of the requirements of the game is enhanced, this system will be altered to meet these changing requirements. This system was much simpler to set up than the shortcut reader since I already had an idea of what I was going to do, and I did not require anything more complex than <code>eval()</code>. The script essentially reads a file containing a python list, then runs an <code>eval()</code> on the <code>open()</code>ed contents of the given file, thus loading the list into the script. From there, the <code>main_loop()</code> function reads and <code>.place()</code>s every <code>Element()</code>. This is a major milestone as every level will likely use this mechanic to load screen elements. This system also may allow me to eventually make a visual level editor (which will tremendously increase the pace at which I develop the</p>	<p><a href="#">Shortcut reader code</a>  <a href="#">Example level config file</a>  <a href="#">ISO 8601 - Wikipedia</a></p>

	<p>game) and will most likely allow user-made levels to be loaded into the game. The significance of this concept to the design of my code prompted me to shift to Alpha 1.</p> <p>Finally, I updated the documentation of my project, improving the changelog to be more readable, and switching to a YYYY-MM-DD format to align with ISO 8601.</p>	
2022/01/01 to 2022/02/06	<p>I had planned to do more work on the project during the holidays, but most of my effort was utilised in making unrelated python scripts such as integration or WAVE file generation. The latter may be used later for the generation of game music and SFX.</p> <p>Over this period, I also revised some incomplete scripts, such as a button concept (originally abandoned a long time ago), which will possibly be used as the basis of the GUI menus.</p> <p>I updated some commenting on the <a href="#">classes and functions script</a>, and added in a <code>Timer</code> class, which can track the time since its creation, and be reset to a new start time. This class will likely be used to set the activation cooldown of <code>Button</code>, preventing rapid feedback from a single click.</p> <p>I started work on a new version of the <code>Player (Element)</code> class, which will utilise the updated set of <code>Element</code> class functions and allow the user to move around.</p>	
2022-02-07 to 2022-02-13	<p>I did some further work on the music creation concept, making a script that converts an input tuple into a WAVE file. - This was used to create renditions of Star Spangled Banner using the three variants. I hope to one day expand this such that prewritten sound files can be generated on startup, minimising the portable size of the project. This implementation was easier than I expected as I had accidentally made the script future-proof.</p> <p>More relevantly, I returned to working on the pygame visual elements. I made the game window resizable, which was very easy, but then I attempted to make the elements scale with the screen, which was very challenging. I had significant difficulties with this, since the elements all scaled to the wrong size, for some reason following a parabolic curve of offset. I was unable to fix this problem, so I dropped it for now.</p> <p>I also made a significant change to the level config file format, changing it from a list of elements to a dictionary. This means that the keys of the dictionary represent the different aspects of the level, and this also allows hidden debug entries, and this will make level config files much more readable as well as writable.</p> <p>Finally, I improved upon the graphing in the "luck-based" concept, adding a matplotlib.pyplot graph to better visualise the distribution.</p>	<p>The first half of the graph, Number rolled against frequency of number</p> 
2022-02-14 to 2022-02-20	<p>I decided to leave the pygame visuals alone for now, since I was not making much progress. I decided to focus more on logical implementations, in order to better understand the required structure for my game. In lieu of a solid idea, I will temporarily work on implementing the logic for an UNO game. This will give me some direction and allow me to escape the trap of wasting time on a non-functional extra function.</p>	

2022-02-21 to 2022-02-27	<p>I worked more on the cards and their display. This was easy and worked with minimal struggle. I then added power cards to my game (excluding reverse). This was challenging as I had to rewrite the existing system to accommodate wild and draw cards. Since the draw two and draw four cards use the same system, I had to differentiate them and also track the current draw amount from stacking. In the end, I got it to work as I wanted it to, using a second returned parameter that specifically denotes the amount to draw.</p> <p>In more hopeful news, I have now decided on a solid idea for my game: a game where you collect keys for your in-game keyboard so that you can input those keys in real life. The interim name was Out of Control (idk why), but I have now decided on the name Keyless Karlsson. In order to move with the arrow keys, the arrow keys must be collected in the game. This also applies to other keys. In the end, after you collect the keys, you will have to type in a password to open a door and win the game.</p> <p>I finally got around to implementing the pygame.mixer module to replace the playsound and threading modules, which is great since it's both a lot cleaner and a lot less hassle.</p> <p>I updated the Player.controls() function to accommodate screen wrapping.</p> <p>I created sprites for my game character, and I have decided on the name Keyless Karl.</p> <p>I started on the keyless logic of the game. This involves locking a key's input until its digital counterpart is collected. To achieve this, I copied the Player.controls() function and modified it to include the found lambda function (found=lambda x: x in found_keys). I implemented the Player object and movement, but have not implemented the collectable keys yet. The player is successfully limited to only the keys in the found_keys list, which is good. Testing of this feature was successful.</p> <p>Overall, this transition to an actual game idea was a lot smoother than I expected, I stored the Uno concept away for later, I might implement it into my game, but do not expect to.</p>	
2022-02-28 to 2022-03-06	<p>I added sprites for the keys, and implemented them. This includes an Element.rect() that returns the range of coordinates that an element occupies, such that when the Player.rect() intersects with a Key.rect(), it triggers Key.rect_check()'s positive. This adds the key to the found_keys list and removes itself from the list of elements.</p> <p>I added clicking detection (unsure of usage for now), and optimised lots of code. In particular, I cleaned up the code relating to the Timer class, making the implementation smoother. I added all the directional keys and set up a basis for an anim_loop() function, in order to have a collection animation, and I fixed a bug where hitboxes (logically) were vertically flipped away from the sprite, which only really altered the behaviour at the top and bottom of the screen.</p>	
2022-03-07 to 2022-03-13	<p>I implemented a direct method for playing background music and added some placeholder music to test it out. I then updated the Karl sprites to be aesthetically <del>worse</del> better, since they were very off-putting previously,</p> <p>I removed the Position class for now, as it was orphaned, with regular tuples instead being used to store coordinates, however, if I can get around to rewriting the code, I will be able to reimplement this class. I added a very roughly made rainbow RGB cyler function to cycle the background slowly during the key</p>	

	<p>collect animation. The rect() and place() functions were both broken/using dead code, but I cleaned this up and restored their functionalities. I also did some slight overall code and comments cleaning up and file shuffling.</p>	
<p>2022-03-14 to 2022-03-20</p>	<p>This week was very productive, as I renamed the repo to Keyless Karlsson, since I have abandoned the card game idea, and I merged all game loops and functions under a master Game class. This will greatly improve intercommunication across the game functions, as well as adding further structure to the code. I also updated the readme to be not about card games, which I forgot to do previously.</p> <p>FLIPPERS! I added the flippers finally. In order to work with horizontal flipping, I changed Karl's sprites slightly to visually indicate right and left. Originally, this was just a green and red recolour, but I changed this to an arrow to not have the sprite look bad. As part of the function of flippers, I added a cooldown period of 10 seconds, with an accompanying charging animation to indicate when the cooldown is finished. I also implemented a rough and incomplete version of blocking, such that Karl cannot walk across the flipper. I will improve this function at a later date.</p> <p>Overall, the work done this week will aid me greatly across the next stage of my project, as I design the levels and settle into the finer details of the challenges presented within the game.</p>	
<p>2022-03-22 to 2022-03-27</p>	<p>I realised that the flippers were simply not functioning properly, acting as one-way paths rather than as solid objects, so I spent a significantly longer than expected time debugging and fixing them. This extended time was due partially to my desire to hold onto the existing code rather than completely redesign the flippers, which is what I ultimately did anyways. Given that we have exams next week, I do not plan to work on my project next week, and I may possibly extend that break until the last week of the holidays, as this may give me the time I need to rethink the project, and also I plan to work on the visual and multimedia assets of the game in this time.</p>	
<p>2022-04-24 to 2022-04-27</p>	<p>As I previously said I would, I took a break from the main project code, but in the past month, I fiddled with the .wav generator, and experimented with some music concepts.</p> <p>I gutted my entire project, and made a long list of functions to rewrite and completely remake. This rewrite gives me the chance to reimplement the `Position` class :)</p> <p>I forked the repository and created a "rewrite" branch. I will merge this into main when I get things to work. I started rewriting the `Element` class, implementing the `Position` class and a new `Vector` class. Vector is used to track the movement of an element from a given Position. As a result of adding these two classes, I was able to remove some functions that were obsoleted by them, making the code overall cleaner. I also rewrote the rect method to return two `Position`s rather than a `numpy.linspace()`.</p> <p>The blocking attribute is now called "solid"</p> <p>Overall, the actual main.py functions still need to be rewritten for the new format, and the template functions need to be completed and cleaned up, but I have made significant progress towards a cleaner,</p>	<p><a href="#">Audacity</a></p>

	<p>more structured, and overall better project. As of yet, it is not at all able to be run, since it is still in fragments, but I hope stitch it together sometime within the next week.</p> <p>For my project documentation, with the impending Part A deadline, I revisited my wav generators and music concepts, using the abandoned sequences to create a passable main menu theme. This combines the sine and saw waveforms, using the latter as the melody, and the former as a backing drone. I was pleased with how this turned out, as I did not expect much, and I got an ok tune with not much effort, largely due to the effort I had previously put into the abandoned .wav files. I used Audacity to combine the two wav files and tune the dynamics of both to an acceptable level. I checked over my DFD level 1 and 2, and realised that I could wrap the Audio and the Visual layer together into one A/V process, since both are called during the same stage of the main game loop. I also cleaned up some stray/misformed arrows and made the labels more readable and make more sense. My Gantt chart was all good. I added a sprite for the unimplemented waterball shooting mechanic, in preparation for its future implementation. I polished up my logbook (and wrote about it in my logbook). I will likely not dive right into coding from this point, except possibly to finish up remaking all the functions.</p>	
2022-05-08 to 2022-05-11	<p>I did not feel like doing proper coding this week, but I decided to do some small touch-ups to the existent code. I also worked a bit more on reunifying the new system and the old subroutines. For Vectors, I was trying to use arctan and cases, but ultimately I fell back to complex and cmath instead. since complex inherently calculates these functions, and is therefore much easier and cleaner to use, as well as actually functional. Now that Vectors work, I need to work on hitboxes, so I can visualise rect collisions.</p>	
2022-05-12 to 2022-05-18	<p>I worked more on Vectors, making them more standard as part of the new world order, and I think I fixed rect collisions. This was rather simple, and thus did not pose much of a challenge.</p>	
2022-05-19 to 2022-05-26	<p>I changed rects to be drawn lines rather than placed marker Elements, making it much easier to see the boundaries. Blocking is now fully covered, as part of the debugging of in_rect(), so that all corners are checked. The in_rect() now checks corners both ways, and thus may end up decreasing the overall speed of the game, but at the benefit of less buggy gameplay. I later went back to the blocking system and overhauled the coordinates calls, since they were still running off old system, but have now been patched to work with the .x .y system rather than [0] [1] system.</p> <p>I added the level loader system in (same as the one made in late December) (well not really implemented yet it's very very broken, since I need to change Game object to interpret the config data). In the old system, Game is initialised with all the information provided already, so I have to rewrite it to be able to accept a string and eval() it, and then somehow gain information about the system from this. I am currently leaning towards an array of dictionaries, but I am somewhat uncertain on if this is wise.</p> <p>I have been experiencing a persistent and infuriating bug where the Player simply does not interact with flippers and keys right now, but I will endeavour to fix this very soon.</p>	

<p>2022-05-27 to 2022-05-03</p>	<p>fixed the collision problem, it turns out it was because the keys were bound to the wrong Player, now fixed so collision works, speed is reduced to proper speed, and also added room movement so now where if room pos within defined rooms then move to next room otherwise stay in same room but go to other side. need to do levels now</p> <p>Work for yesterday and today. Added some testing concepts with levels, changed background temporarily, added more keys, barebones prototype currently. Will now progress to designing levels.</p> <p>ADDED WALLS it was a lot easier than I expected, I guess that's because I redesigned collision and placement so extensively that it was easy to slide in. Walls longer than the player must be at least (player speed) thick to always work, since thinner walls can be clipped through by the player's corners not being in the wall dimensions. I now plan to make walls toggleable, and to add recharging back to flippers.</p> <p>Firstly, I added two new base classes Interactive(Element) and Blocker(Interactive), for player rect collision and player movement blocking respectively. Key inherits from the former, and Wall and Flip inherit from the latter. I also fixed Flip not recharging, so that it recharged, and I added a toggle to Wall that disables its rect_check(), allowing the player to walk through it. I also added boosters, which allow movement in the direction they face. The player is boosted in that direction by the magnitude of the boost multiplied by the player's movement speed. The last piece of the puzzle I needed now was the buttons, after which I could begin assembly and (probably not) testing. I started with Trigger (renamed button) sprites.</p> <p>BIG BRAIN TIME I added a dictionary of starting conditions to replace the generic starting condition and this allowed me to start in any defined starting state, with keys, flipping, position, and room defined. I did more work on designing levels, almost entire game was complete yesterday, albeit with significantly less puzzle than desired. Needed to now get music, boost sounds, and fill out marking criteria stuffs.</p> <p>Today I added more levels to incomplete levels, decided to strategically abandon placeholder level(s?), and added more interconnectedness between the levels.</p> <p>In other news, I copied over the ATM batch file loading system, and added the module wheel files to allow running on systems without these modules or internet.</p> <p>Since the project is due today, I am very much sweating documentation, though I have considered doing significantly less commenting than planned.</p>	
-------------------------------------	---	--