Programming has been a passion of mine since elementary school. I thought then that I would grow up to be a game developer. Over the years, I have taken several programming classes, which have taught me Java, JavaScript, HTML, CSS, Python, and C++. While I am not the most talented developer, I can hold my own in the above languages. Currently, most of the programming I engage in is for the microcontrollers utilized by my Ground Systems team. I also use Python for data analysis after we conduct propulsion tests. In my free time, I work on personal projects, such as developing this website. I am very interested in machine learning and artificial intelligence and hope to gain experience working on projects that utilize both.

One of my classes featured a challenging final project that tasked me with creating a Pac-Man-style arcade game entirely from scratch. The resulting game is a tick-based adventure where the player assumes the role of a miner on a quest to discover all the hidden barrels of oil within the map. At the commencement of each level, gold and oil are dispersed randomly beneath the pink earth. Additional game elements are added spontaneously after a minimum number of ticks. Various items are uncovered as the player mines and added to their inventory. The player can utilize these items during the level or save them for more challenging levels. Amid the quest for buried oil, the player must contend with default and advanced enemies, who both can pursue the player. When the player is within direct line of sight, these enemies will chase them. Advanced enemies also utilize a queue-based maze algorithm to track the player when within a predefined set of legal moves.

This complex project featured numerous dynamic components and intricate rules, posing challenges in organization and management. With hundreds of dynamically allocated game objects, each potentially deleted during a tick, I needed to be creative in structuring my classes, implementing inheritance, and optimizing object storage. Simplifying the process, I identified shared characteristics among game objects and grouped them under a unified "Actor" base class. Collectible objects were categorized under a "Treasure" base class, while the main character and enemies fell under a "Human" base class. This hierarchical grouping continued until the lowest-level derived classes contained unique code. The "GameWorld" class maintained a vector of pointers to each object, and I implemented a function to update this vector during each tick, handling the addition and removal of objects. Interaction between classes was intricate, requiring careful design to avoid violating encapsulation. I heavily relied on accessor and mutator functions, defined at the highest class hierarchy level, enabling, for example, the "GameWorld" class to update the player's inventory, exemplifying the intricate interplay of various elements.