**Introduction**

Syobon no Akushon (a.k.a. Cat Mario) is a Mario-style adventure game developed by Chiku which features numerous unorthodox traps that are intentionally placed to kill the character by surprise and cause extreme frustration. The goal of this project is to develop an AI that can learn to control the agent (Cat) to get through the horizontal game level by evolving after numerous deaths. We implemented the project using the NeuroEvolution of Augmenting Topologies (NEAT) method developed by Kenneth O. Stanley and Risto Miikkulainen, and developed our own Game Capture and Training Environment to monitor the training progress.

**Game Capture and Training Environment**

Without direct access to Cat Mario's in-game data, we utilized computer vision (OpenCV) to extract useful game information and convert them into low dimensional training data. A complete training environment was also developed to monitor and record the performances of each genome in each generation. The key features of the Game Capture and Training Environment are:

1. Utilize Windows API to efficiently capture the real-time game video feed for analysis
2. Perform agent (Cat) detection using template matching and extract its local view to eliminate itself from output domain.
3. Classify local view image into 13 * 13 blocks of three possible values: -1, 0 and 1, which represent hostiles, background and standing block respectively and they are visualized as black, grey and white blocks.(see Figure 2)

**Methodology**

NeuroEvolution of Augmenting Topologies (NEAT) was selected as the algorithm for the project. This neuro-evolution algorithm is a genetic algorithm that selects and evolves both the structures and weights of a population of neural networks (NN). Each genome is encoded by an NN (see Figure 4) with fitness measured by its performance (horizontal distance travelled) of running the Cat Mario game. At the beginning of the learning process, the population is initialized with fully connected networks with 169 input nodes (1 for each block of game window) and 4 output nodes (4 keystrokes). The connection weights are randomized and there are no hidden nodes for initial networks for minimizing structures throughout the evolution process. The genomes are clustered into species based on the compatibility between genomes` NN structure:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot W.$$  

where E, D are the number of excess and distinct genes and W is the mean weight difference

The evolutionary processes for each generation are summarized as below:

1. Evaluate the fitness for each genome (NN) in the population by game runs.
2. Assign quota of offspring for each species based on species fitness, which equals to member genomes’ mean fitness. Species with higher species fitness will receive smaller quota to ensure diversity of NN structures in the population. Genomes with least fitness are eliminated.
3. Produce offspring from each species by crossover and structural mutation. Crossover inherited the genes from the parents (see Figure 5). Structural mutations include add-node and add-connection mutations (see Figure 6). Both are created within current structure for minimized initial mutation effect.
4. Assign newborn offspring to current species using the compatibility formula. Weight mutation would be performed on the entire population. Each weight either perturbs or not.

**Experiments and Results**

We evaluate Cat Mario AI by examining the population’s fitness distribution over several generations. Our goal is to find a set of parameters such that the population can converge faster, faster and has genomes that reach a distant location in early generations.

1. The experiment above demonstrates the AI’s ability to find paths and get around a major trap after evolving another 54 generations
2. Tuning parameters: Two mean and max fitness over generation graph on the right illustrate the impact of an important parameter: max_stagnation, which is the number of generations a species can survive without improvement. Using a low max_stagnation value can typically converge sooner because it aggressively eliminates poor performers. However, it discourages randomness in new species that potentially the chance of finding a better solution.
3. Some of the major challenges in this project include having a number of parameters in NEAT to fine tune, having a slow training speed (limited by game speed), having to interface the game with Computer Vision, and building a user friendly training environment.

---

**Figure 1** Original Game Window  
**Figure 2** Local View Block Classification  
**Figure 3** Training Environment (Game Level Map)  
1. Implemented simultaneous localization and mapping (SLAM) that maps out the entire game level while tracking the positions of the agent (Cat) and game window in the topological game level map in real time (see Figure 3)  
2. The trajectory and the maximum distance travelled by each genome (Cat) in a generation are marked on the map for statistical performance analysis  
3. The system recognizes game events such as Cat’s death or Cat getting trapped which will appropriately reset the game and move on to the next genome.