P6 Writeup

We do not intend to participate in the competition

Changes to the template:

- Changed the order of the options provided for the level for better access later on in the code.

In Individual_Grid

- Changed mutate. We added constraints to make the level more playable.
    - Constrained air to be the majority of the level at 92% from lines 6-12 (the middle of the level) and the rest would be a random selection of blocks and coins with various probabilities. For example, we generated question blocks randomly for 2% because too many question blocks would lead to too much reward. These blocks were constrained to not have blocks on top of them and to generate in multiples at a time.
    - Generated pipes of random length and enemies between lines 12-16(bottom of the map). We made sure that the pipes have their tops and would stay on the floor.
    - Made sure that lines 1-5 would be all empty spaces for aesthetic purposes because mario would not be able to reach there.
    - Made 'holes' on the floor 1% of the time using empty spaces to account for better playing experience.
- Changed generate_children take the genomes from self, using uniform crossover., and producing one child
- Changed random_individual to do more or less the same as empty_individual with a random generator.

In Individual_DE

- Added some fitness constraints. Instead of penalizing, we rewarded for it being in the correct constraint, so the genomes that are in the correct constraint will have more fitness, this having a higher chance to be chosen.
- Changed mutate – Originally, mutate would only make alterations with a 10% chance, but we changed it to always mutate. We also created max widths for platforms and holes to ensure that the level is playable and there aren't holes that are too big to jump over. We also limited the height of pipes to a reasonable amount for Mario to jump over.
- Implemented 2 genome selections to be used in generate_successor. We implemented roulette_selection and tournament_selection for each of the children that came from generate_children
    - Roulette selection were made to loop over the individuals in the population, then randomly select an individual. Those with a higher fitness will have a higher chance of being selected. However, those with a lower fitness will still have a chance of being selected
    - Tournament selection were made to select 40% of the individuals from the population and select the one with the best fitness. There might be a chance that there exist an individual in the other 60% but ultimately, we still get the individual with the best fitness in the given group.

- We also modified the population to 200 to ensure that the run time is optimal (not too slow) and not lower because we wanted to also have a large population.

How things work in the code:

In Individual_DE

Mutate

- 10% of the time mutate is called, each DE will be slightly shifted on both axes. For certain DEs with special properties like pipes, platforms, and holes, the widths and heights will be offset. Block types are also flipped in the case of blocks being breakable or not.

Generate_children

- The code selects two different indices and slices a sublist from each parent to combine them to form the new genome to be mutated under one parent. For the second parent, the indices used to slice the list are switched so that the sublists from each parent are slightly different from the first new genome. This second new genome is then used for mutation under the second parent. These two new parents will be mutated and a single Individual_DE child will be returned.

Favorite Level

In Grid

- We like this level because it gave us more freedom to change and it looks much better than in DE
- It took us 30 generations to produce this level. Our average generation time for this was 28.282754710742406 seconds.

In DE

- We like this level because of how simple it was to implement and how it produced good looking structures. Special parameters were easy to control like the height and width of things.
- It took 7 generations and our average generation time was 21.588624715805054 seconds.