

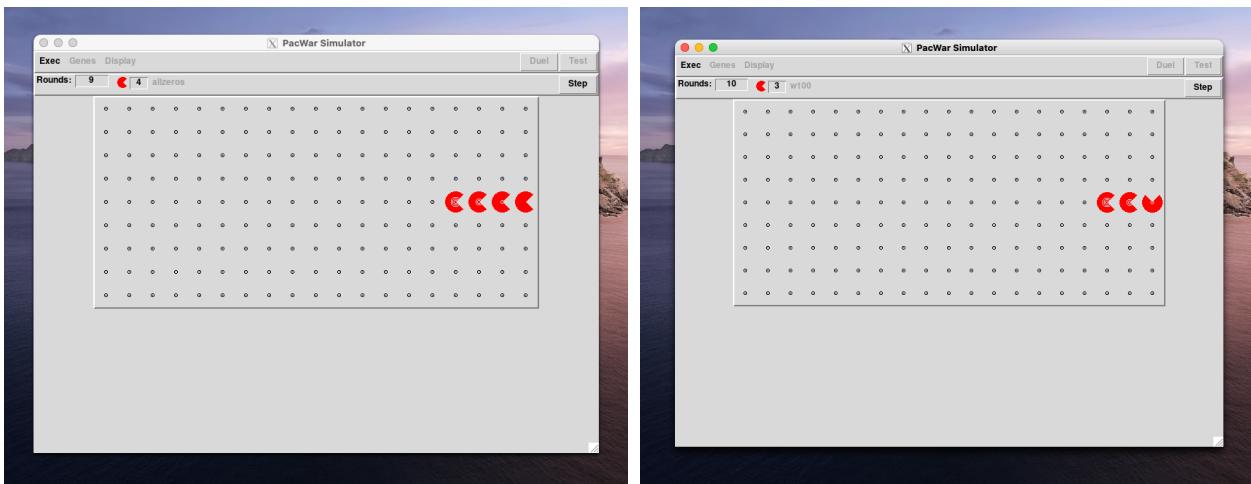
PacWar Project
Min Hung Shih (ms198), Chinmayee Schandra (cs124)

Understand the rules of Pac-World

1. Look at the rules that define the behavior of a gene (the different components) and make sure you understand them. Use the simulator to understand the impact of changing the U, V, W, etc genes.e

Initially, we assumed that the movement on the board is a result of Pac mites moving, which is incorrect. The actual mechanism is that the Pacs don't "Move between cells", instead it either gives birth to a new child, turns, or dies. Take W for example, when we set W as [1, 0, 0], it means that if an aged 0 pac is facing the wall, it will turn 90 degrees counterclockwise and age 1 year old. As for U, it dictates how the mother pac mite gives birth to the new one. For instance, say U is [0100], this means when a year old Pac mom is facing an empty cell, it creates a child that's gonna turn 90 degrees counterclockwise from its parent's direction.

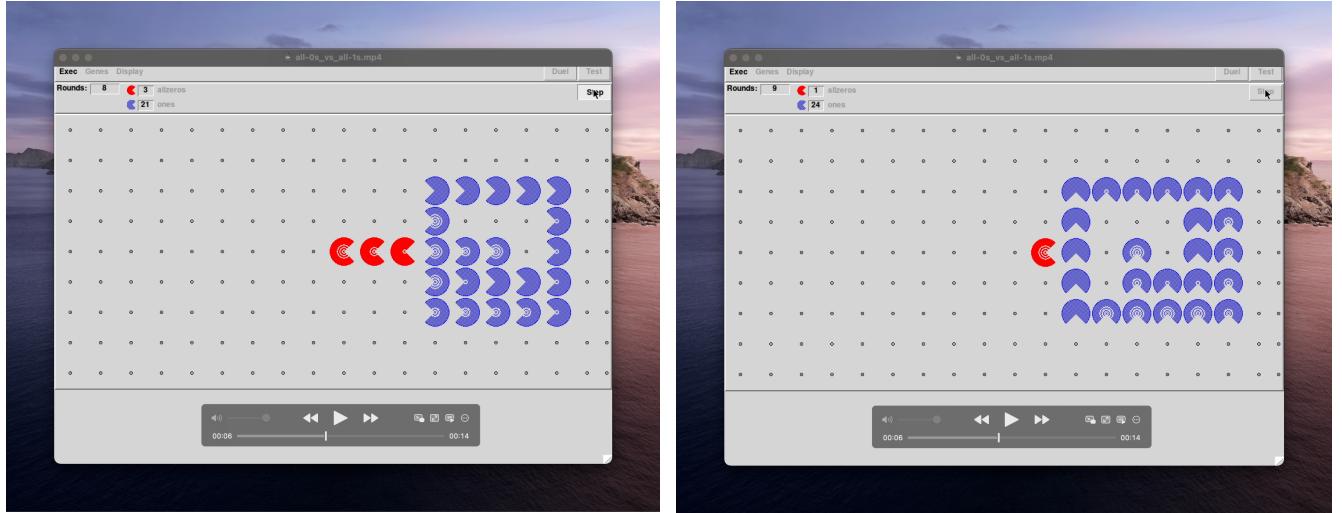
In summary, the gene sequence is divided into subgroups and each group encodes some rules for the pacmites.



2. How does the all-zeros gene behave when confronting the all-ones or the all-threes genes? Make sure you understand why that behavior results starting from the given rules.

All zeros collectively just went straight into its opponent's battlefield. On the other hand, each mite continues to change its direction then gives birth to its child, therefore creating a rectangular frontier pushing outwards.

When all 0s confront all 1s, since the 0s mite is blocked by the 1s, the ones at the end of the line continue to die off because of reaching maximum age. In addition, when the 1s turn to the opposite direction of 0s, they further expand out the frontier, which later traps the remaining 0s in it.

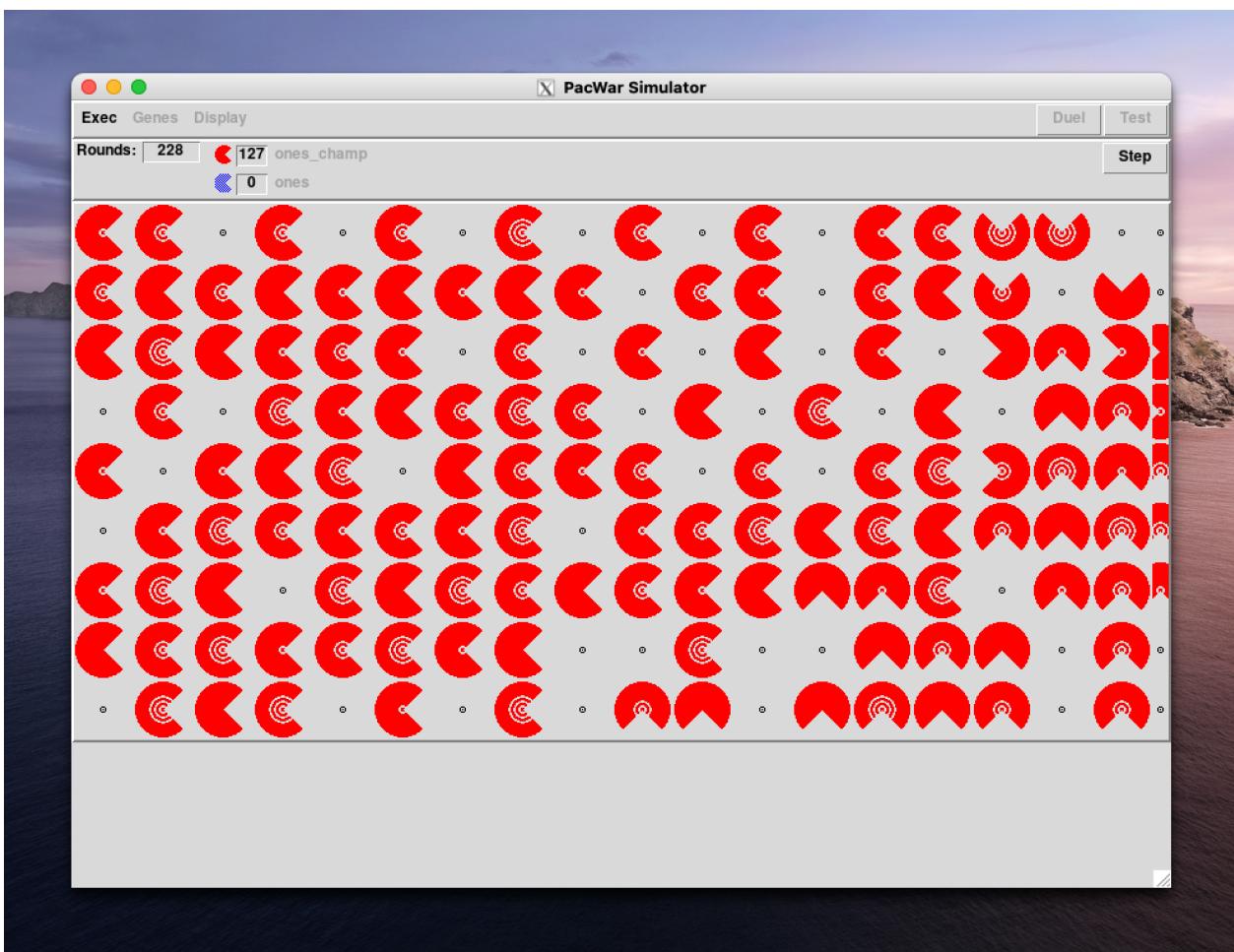


Use the simulator for hand exploration of the space

1. Based on the rules alone, can you come up with good settings for the different gene positions? Try making variations of the all-ones and all-threes, and pit them against the all-ones and all-threes. Can you find a variant of all-ones that beats the all-ones? The all-threes?

Modified ones (alpha) V.S All threes

Modified ones(beta) V.S All ones



Modified ones(final) V.S All ones

Computationally explore the space of genes

1. Which of the algorithms we have studied in class appear relevant to the problem of finding a killer gene?

From what we've seen trying to randomly modify and generate solutions, we're inclined to select genetic algorithms. There are several reasons:

- It's easy to generate a solution that can beat all 1s or all 3s by just modifying a few genes from themselves (4-5 genes in our test run provided previously). This seems to indicate that if we choose a good enough gene to start from, it can lead to acceptable results. Thus, we require the algorithm to generate variations by mutations and select good variations randomly. (Hill climbing can also achieve this, so we're also looking into this)
 - For this problem, it seems to us that it isn't possible to find the global optimal solution. In order to prevent ourselves from reaching a local optimum and looking for as many hills(local optima) as possible, we expect the algorithm to explore different parts of the solution space. We are thinking about adopting the Genetic algorithms crossover technique to accomplish this.

2. Implement the simplest among them and use it to find a gene that all beats all-ones and all-threes.

- Randomly pick 5 positions on the gene sequence then flip it to 0 (Mutation)
 - It also works when I flip it to values from 0-3; however, the success rate isn't that high and we see many draws.

```

random.seed(time.time)
rand_idx_to_flip = []
for _ in range(5):
    rand_idx_to_flip.append(random.randint(0,49))
ones = [1] * 50
for idx_to_flip in rand_idx_to_flip:
    # ones[idx_to_flip] = random.randint(0,3)
    ones[idx_to_flip] = 0

```

3. Once you have beaten the all-ones and all-threes, how do you find even better genes?

To find better genes, we have a few ideas:

- Mutate the current genes, with a mutation rate less than 0.02 (Currently we think we flip too many positions, also we look to see if there's a heuristic for guiding which position we should flip)
- Also, add criteria for choosing the winning genes. For example, the winning genes have to win in a specific number of rounds.

4. What is your plan to find genes that will win the tournament at the end of the term?

Our preliminary plan is to generate a large enough pool of genes. After that, let them battle to filter out some unusable genes. After a few rounds for us to obtain a small enough size of genes (~50), mutate from these genes randomly by 1-3 genes (perhaps selective, meaning we only mutate at the U or V genes) and create their doppelgangers. Have a battle between the 50 genes that survived and their doppelgangers. For the losers, crossover them with themselves at a set position then continue to re-battle in this losing group to find the winning genes in the losing group. Try out the winning & losing group's champion against 1s and 3s.