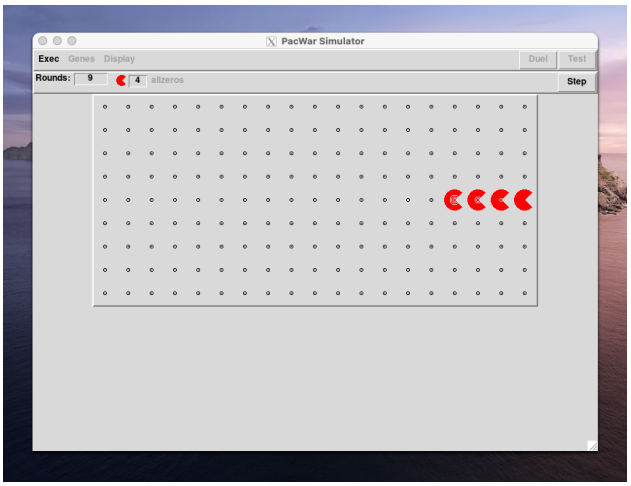
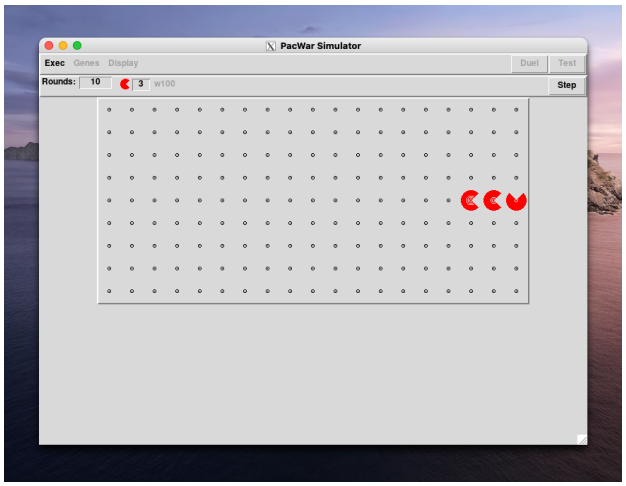
PacWar Project
Min Hung Shih (ms198), Chinmayee Schandra (cs124)

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

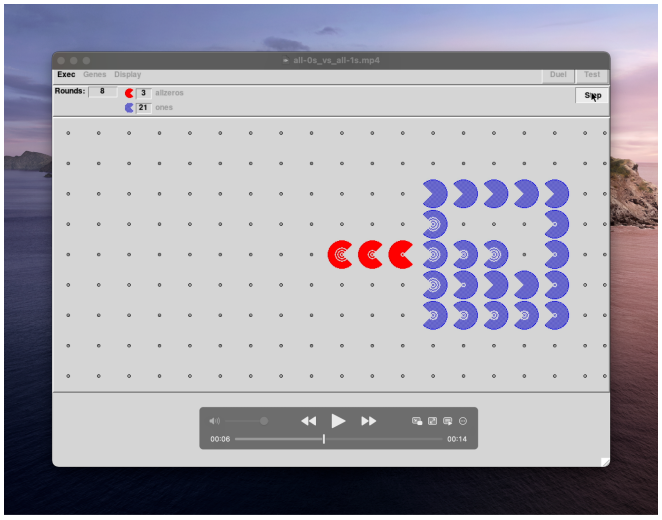
Initially, we assumed that the movement on the board is a result of Pac mites moving, which is incorrect. The actual mechanism is that the Pacs don't "Move between cells", instead it either gives birth to a new child, turns, or dies. Take W for example, when we set W as [1, 0, 0], it means that if an 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.

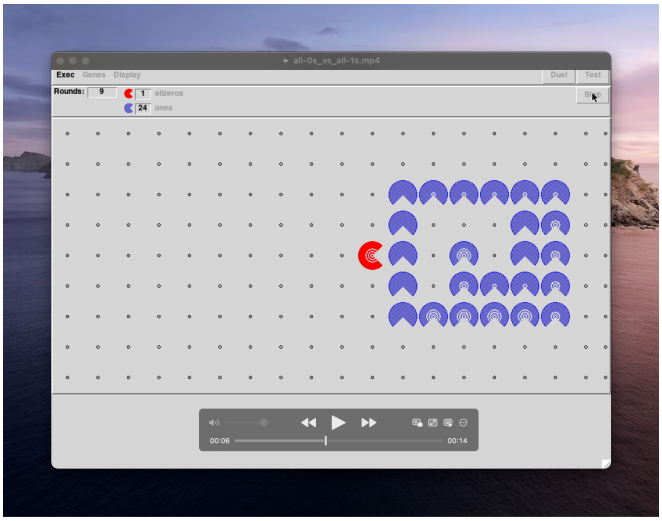




2. How does the all-zeros gene behave when confronting the all-ones or the all-threes genes? Make sure you understand why that behavior results starting from the given rules.

All zeros collectively just went straight into its opponent's battlefront. 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.





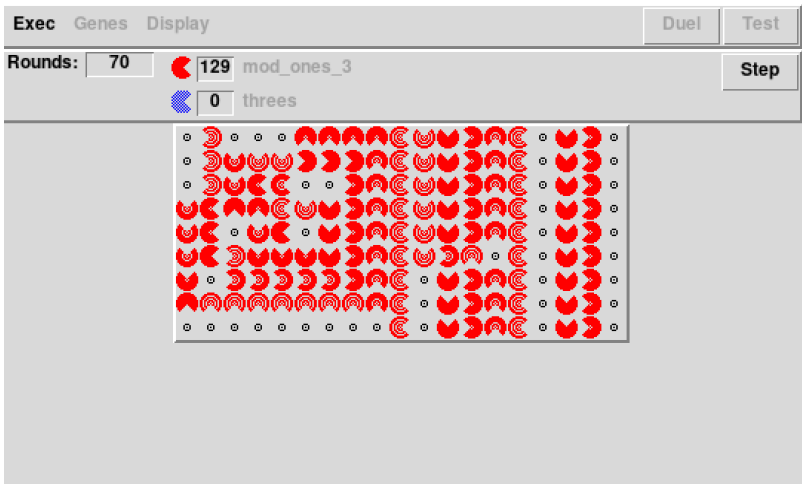
Use the simulator for hand exploration of the space

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(final) V.S All ones & All threes

- Beating both All 1s and All 3s
    - Beat All 1s in 181 rounds compared to 246 rounds previously
    - Beat All 3s in 70 rounds which is significantly better than previous result
- **Genes**: **[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]**

```
Number of rounds: 181
Mod-Ones PAC-mites remaining: 134
Ones PAC-mites remaining: 0
Number of rounds: 70
Mod-Ones PAC-mites remaining: 129
Threes PAC-mites remaining: 0
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```



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 considering 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 3 positions on the gene sequence then flip it to 0 (Mutation)
- Previously we choose 5 positions, though the success rate is higher, meaning that the chance of beating All 1s and All 3s are higher, the rounds it took is generally higher compared to when we only flip 3 positions. For instance, it only takes 70 rounds to beat all 3s compared to over 200 rounds last time.

```
random.seed(time.time)
rand_idx_to_flip = []
for _ in range(3):
    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 improve on the current result, we have a few ideas:

- Setting the mutation rate between 0.02~0.06. Since there are 50 genes, we expect to have 1-3 genes being flipped.
- Also, add criteria for choosing the winning genes. For example, the winning genes have to win in a specific number of rounds. Meaning we abandon those solutions

that take more than a certain number of rounds. This should aid in our process of searching for the optimal solution since besides the final result, the rounds it takes are also a qualitative measurement of how good the genes are.

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

Generating population of genes

Our preliminary plan is to generate a large enough pool of genes. After that, let them battle to filter out some unusable genes. Filter out those losing to All 1s and All 3s.

Battle and Mutations

After a few rounds for us to obtain a small enough size of genes(~50), mutate from these genes randomly by 1-3 genes selectively, specifically we want to divide these 50 genes randomly into 5 groups, and each group will have mutations on U,V...positions separately. Based on these mutations, we can create their doppelgangers.

Genetic Algorithm

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, but instead of all 1s and all 3s, we were think about adding 5 random pacsites into them so our genes don't actually only know how to beat the all 1s and all 3s but also are capable of facing different opponent.