# Plants vs. Zombies

### Project Introduction

Our system is to design a little circumstance that based on the game Plants vs. Zombies. A zombie who wants to eat a brain being born randomly in a *10x10* map. However, there are also spikeweeds on the border of the map. There are also 50 brains distributing in the map randomly. The zombie will lose 5 points when it walks on the spikeweeds and loses 1 point when it chooses to eat in a place without the brain. On the contrary, it will get 10 points when it eats the brain successfully. The zombie's actions include moving up, moving down, moving left, moving right, eating brain and keeping still. It can know the situations of its surroundings. One zombie can only move 200 steps. Our project is to find the best zombie who can get the highest score after a huge amount of times' evolving.

### 2. Program Detail

## 2.1 Genotype

In this program, the genotype is the digital representation of the environment in which each step of the zombie is located and the state around it. According to the rules of the game, there are 5 squares in the surrounding area of the zombie, and there are 3 kinds of states in each square: there are has brains, has the wall, and nothing. Therefore, the total state has 3^5=243 kinds. Therefore, we use the hexadecimal 5-digit number to express the position information of the zombie systematically. In the program, genne[] is a byte array. The index of the array represents the genotype.

## 2.2 Expression

In this program, the expression is action Code which makes the gene[] change to action()(zombies change position on the map)

## 2.3 Phenotype:

In this program, the phenotype is represented by a trajectory of a zombie moving on a map, that is, a collection of 200-step movements. These moves are real-life actions that achieve a series of movements of zombies on the map through state changes.

## 2.4 Fitness：

In this program, fitness is expressed as the score of the zombie, expressed by a fitnessCalc function. According to the rules of the game: Zombies successfully eat the brain +10, move or not move +0, send out the action of eating the brain but eat nothing -1, hit the ground thorn -5. There is a total score after each route ends. The algorithm can calculate the average score and the highest score of the population.

In addition, in order to keep the most ‘intelligent’ zombies evolved, the zombies will randomly move 200 steps in a 10\*10 grid and perform 1000 simulations on the result. The zombies must have good performance in this thousand games. To get the ultimate high score, instead of an occasional big hit.

## 2.5 Population

In this program, the population means 1000 zombies initially placed(seed = 1000). During the experiment, the total number of individuals included in the population does not change (always 1000)

## 2.6 Individual

In this program, an individual means a single zombie. The zombies have random actions including up, down, left, right, not moving and eating brains. The zombies will randomly move 200 times in a 10\*10 grid.

Have an array of genes (record path encoding), whether to survive, generate genes, getActionCode (path encoding by the state to determine Action), etc.

## 2.7 nature selection & evolve

In this program, the natural selection method is composed of three parts: survival, crossover, and mutation.

2.7.1 Survival

In this program, the elimination method selects the “roulette” method to probabilistically eliminate individuals in the population. That is, an individual whose final score is lower than the average score of the population has a 70% probability of being eliminated (that is, the individual ‘isAlive’ attribute is marked as false) and the individual whose final score is higher than the average score is not absolutely safe, and the individual still has a 30% probability being eliminated.

The reason for this is to not limit the maximum value of the algorithm to certain genes: for example, zombies sometimes pass the brains without eating, and if such genes are eliminated, it may be good for the current maximum growth, but It's easy to limit the results, and after running all the functions, you may not get the true best results of the situation.

In addition, after the process is completed, nearly half of the individuals will die in the entire population.

2.7.2 Crossover

When the elimination process is over, the larvae are mature and can be mated and propagated. Since the total number of population arrays is 1000, what we have to do is replace the individuals that have been eliminated with new individuals by crossing and mutating.

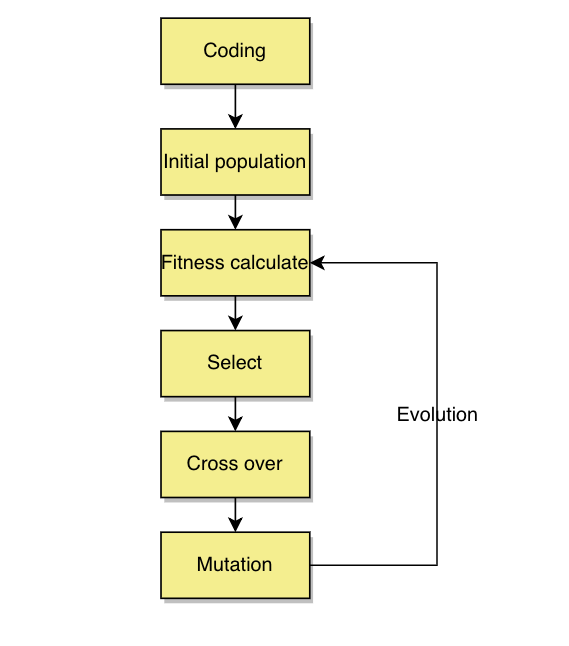
In this program, crossover means: randomly selecting surviving zombie parents to mate - assigning parental gene equality probabilities to offspring. Here we traverse the gene array if the probability is lower than the union distribution, the father's byte is assigned to the child generation, and vice versa. This achieves cross inheritance.

2.7.3 Mutation

During the mating process, the original gene of the offspring has a chance to mutate into a new gene, and the mutation probability we set here is 0.1%. Mutations can limit the extremes of our program to one place (although most mutations are harmful).

In this program, mutation means that the probability of a byte in the progeny gene changes from 0 to 1 or 2. These cases are random and the probability is the set mutation probability.

### 3. Flow Chart



### 4. Configuration Parameters

Initial seed: 1000 — initial population has 1000 zombies individual

The proportion of organisms that survive and breed: 0.5 — that means kill half of the population. Our program achieves this.

Fecundity of mating: 2 — the total number of individual for the population not change.

Generations to reproductive maturity: 1 —after select, the mature stay for 1 generation, and can do the crossover.

The maximum number of generations: 10,000 — we do about 3000 generations. Initial seed: 1000 — initial population has 1000 zombies individual

The proportion of organisms that survive and breed: 0.5 — that means kill half of the population. Our program achieves this.

Fecundity of mating: 2 — the total number of individual for the population not change.

Generations to reproductive maturity: 1 —after select, the mature stay for 1 generation, and can do the crossover.

The maximum number of generations: 10,000 — we do about 3000 generations.

### 5. Test Introduction

## 5.1 TestIsInMap:

Generating a map of 10\*10 for testing the function change. Special points (11,4), (-1,4) and (5,5) were taken for testing. Because the boundary condition is 10\*10, the – 1 or 11 equivalence cannot pass the test, so the result is the same as expected, and the test is successful.

## 5.2 TestHasBrains:

Generating a 10\*10 map to test the feasibility of the function. After the map is created, the test begins directly to test whether the map contains (1,1) points, and the result is false. After that, the map is inserted into all the points and the test is continued. The result can detect (1,1) points, and the test is successful.

## 5.3 TestEatBrains:

A map of 10\*10 is generated and eat brain operation is performed directly. The zombie eats (1,1) points directly and returns false, indicating that the eating-brain action is unsuccessful. After filling the map with 100 brains, the operation of eating (1,1) points is performed again, returning true and eating the brain successfully. Use the HasBrain method to test (1,1) whether there is still Brain, return false, indicating that brain elimination success. Using the HasBrain again to test (2,3) points, see whether there is a brain, return true, indicating that hasBrain function is normal and Zombies have eaten the (1,1) brain, there is no impact on other points.

## 5.4 TestGetStartPoint:

Generating a map of 10\*10, and randomly generate a starting point on the map. Since the starting point is randomly generated, it is not possible to test whether a particular point has been successfully created. This test uses IsInMap to test whether the point is within the scope of the above map and the result is true, indicating that the starting point of random generation is within the scope of the map and meets the functional requirements.

## 5.5 TestSetIndividuals:

Establish a new individual, set the gene of index 2 to 00 for the new individual, and calculate the fitness of the individual after survival. The test results are the same as those predicted by getting fitness, which indicates that the individual was successfully created. The set individual function works well. The test is successful.

## 5.6 TestGetFunctionofState:

Create a state object with a = 1, b = 2, c = 3, d = 4, e = 5 as its five parameters. Each get function is used to get the corresponding data. The data obtained are exactly the same as the set data, indicating that the function is normal and the test is successful.

## 5.7 Test Algorithm

5.7.1. Test kill function:

The aim of kill function is to kill half of the individuals in the population. So we write a test function to count the individual who is alive and dead. So ‘live/dead' individuals should be close to 1(we set parameter between 0.9-1.1)

5.7.2 Test evolve function:

The output individuals of evolving () must all alive! So, we init a population and do evolve(). Then verify if all individuals in this population are alive, return true.

## 6. Results

In the results, I have run the code five times. Here are the results of the results. In these diagrams. We can find the increasing of the fitness is getting slower. In that case, we can draw a conclusion that the fitness will reach 500 if we run enough evolving times. Because the best zombie can eat all of 50 brains without losing points. We can also find there are abrupt changes in the resulting diagram. The reason for that is there are mutations in my algorithm. These mutations are good for the zombie to become better.

There are 243 kinds of states and each state has its own action. All of these things organize a path for a zombie. The population of zombies in the experiments is 1000. That means we have 1000 different kinds of zombies. Every zombie has its own path. In the system, we let them can only walk 200 steps and get their map states and birthplace states. What’s more, there are 1000 simulations in this system. As a result, we can find the zombie who can really adapt to the 1000 different states of map and birthplace.