COMP 9001 Assignment2 Report

Student name: Zedong Wen

SID: 500334863

Section one: Talk about your development process

Q1: How did you handle wraparound for movements and collisions?

To solve the problem of wraparound for movement, I defined a new function (is_correct). This function is used to check whether our x, y and angle are within our expected range. That is, x is between 0 and width, y is between 0 and height, and angle is between 0 and 360. I added “is_correct” after every object movement to solve the problem of wraparound for movement.

As for wraparound of collisions, I first calculate the distance_x and the distance_y of two objects. Due to the wraparound, I need to consider different calculation formulas when calculating. Therefore, when the relative distance_x of two objects is more than half of the width, I consider calculating the wraparound. The formula is self.width-abs(self.x-other.x), otherwise the formula is abs(self.x-other.x). Then, the distance_y is also considered similarly. With the distance_x and the distance_y, we can calculate the actual distance between two objects, to judge whether they collide or not.

Q2: How did you test your file parser (import_state method)?

For this problem, we need to know two important contents of import_state method. One is to correctly "raise" and the other is to correctly import data.

As for "raise", We mainly focus on two errors. One is FileNotFoundError and the other is ValueError. For testing FileNotFoundError, we can use an existing file and a non-existing file, and the non-existing file should be raised error. For the FileNotFoundError test, i used a file that exists and a file that does not exist. the file that does not exist should raise the error. for the ValueError test, there are several situations to consider, such as 1.the file is incomplete, 2. a line does not agree of a key and value pair, 3. a line contains an unexpected key, 4. a value contains an invalid data type. For case 1, to test for errors, I set the file to have only two lines. For case 2, I set 1 and 3 pairs. For case 3, I set "height 900" on the first line. For case 4, there is the first test of type int "100,200.0,400,400" and contains the string type "100.0,200.0,'three',400"

For imported data, I print out all the attributes to see if there are any attributes without imported data, to check the accuracy of the code.

In short, different situations of positive/negative/edge cases should be considered.

Q3: What's the most complex part of your code, and how would you make it more modular? Indicate the file name and line numbers.

The most complicated part of my code is bot's action (player.py at line 12). My decision is to find "The best benefit step". There are two most important parts. One is to calculate the number of steps required by spaceship to shoot asteroids. The other is how to implement the first step after getting the target asteroid. For the former, I set step_num (player.py at line 67), and here I apply the recursion I have learned. Calculate the steps that spaceship should take to shoot an asteroid. And add them together. For another question, I set step_first (player.py at line 121), which is similar to step_num, but will return to the first step I made, such as (thrust, left, right), whose type is Boolean.

Section two: Talk about extending your game

Q1: Suppose some asteroids carry natural fuel reserves. How would you capture the asteroid and implement refuel capability?

As for how to capture asteroids, when asteroids carry natural fuel reserves, we can increase the scores of such asteroids to make them easier to be captured by our bot. For example, the original score of small asteroids is 150. When the fuel of this asteroid is 10, increase the score to 500 (the extra score is 350). In fact, different asteroids may carry different amounts of fuel. For different amounts of fuel, we can give a formula to calculate the extra score, such as extra score = fuel amount *35. And add the extra score to the actual score of asteroids (actual score = original score + extra score).

For implement refuel capability, we need to consider the way to realize the refueling capability. Considering two different situations (1. Fuel can be added after successfully shooting an asteroid carrying fuel; 2. After successfully shooting an asteroid, fuel can only be obtained by reaching the place where the asteroid was destroyed)

Situation 1: In this case, we only need to change the score of the asteroid and add fuel when the bullet collides with the asteroid.

Situation 2: In this case, we can set that the primary goal of spaceship is to reach the coordinates of the asteroid collision. When this coordinate is reached, we will search for another asteroid.

Section three: Talk about your bot's strategy

Q1: Provide a brief overview of your final strategy

My final strategy is "The most benefit step". In my program. I will calculate the number of steps needed to break asteroids in different positions. therefore, each asteroid will generate a number of steps. divide the fraction of the number of steps needed to break the asteroid by the number of steps needed to break it. I will get the benefit of each step. Then select the step with the greatest benefit to run. To reduce the complexity of the calculation, I set the number of steps for asteroids located behind spaceship to infinity (The angle between that two is between 180 - rotation angle and 180 + rotation angle),

which means that I don't consider spaceship turning around to shoot asteroids.

Q2: How does your program use the available data to make decisions?

Using the coordinates of spaceship and the coordinates of the asteroids, you can calculate the angle of spaceship with the asteroids (based on the x-axis), and then subtract the previously calculated angle from the angle of spaceship itself. Get the angle between spaceship and the asteroids. This angle is the angle that needs to be adjusted. Each asteroid will get this angle. Select asteroids that are not within the angle threshold as the target objects. If the asteroids are in the left front of spaceship and do not exceed half of the adjusted angle (for example, config.angle_increment/2), turn left. Turn to the right in reverse. If the angle is within half of the adjusted angle, it remains the same. For thrust, since the target angle is within 90 degrees, thrust will be close to the target asteroid in any case, so it remains thrust (based on whether thrust consumes energy or not). Finally, spaceship will remain stationary to avoid collisions (no thrust and no steering), when the fuel is below the firing threshold or when all existing asteroids have been theoretically processed (some bullets are still flying and have not collided with asteroids).

Q3: How would you adapt your strategy if you knew about the upcoming asteroids?

Similar to my final strategy. When I know about the upcoming asteroids, I will divide the map into many square blocks of equal size. Count the number and scores of asteroids in different blocks. Add the scores of the upcoming asteroids into the block, and the number of these upcoming asteroids is equal to the number of existing asteroids in this block. Calculate the steps required to reach different modules and the scores of this block. Then run my "The most benefit step" strategy.