Rendezvous-birds: Code Explanation

Mayank Ladha (201201019)
Email: 201201019@daiict.ac.in
DA-IICT, Gandhinagar

Madhav Khakhar (201201054)
Email: 201201054@daiict.ac.in
DA-IICT, Gandhinagar

Shivang Kar (2012010160)
Email: 201201060@daiict.ac.in
DA-IICT, Gandhinagar

I. Introduction

For implementing our strategy, we made a number of changes in the code that was given to us. The changes were made in TrajectoryPlanner.java and NaiveAgent.java along-with a new class named "Heuristic.java" was made to implement out strategy. An object of this class is made in NaiveAgent.java and the solve() method of Heuristic.java is called which shoots the pigs according to our implemented strategy. A separate class was made to keep the code simple and understandable. Class-wise code is explained below.

II. Class-wise Explanation

1) **NaiveAgent.java** - An object of the class Heuristic.java is made here and the solve() method of Heuristic.java is called here.

2) **TrajectoryPlanner.java** - In this class, a method similar to estimateLaunchPoint is made with the name "estimateLaunchPointMinimum" which selects the minimum angle out of the two angles to reach the target point. This was required as we needed to shoot with the minimum angle as well as maximum angle depending upon the situation.

3) **Heuristic.java** - This class is created for making all the calculations and taking decisions according to the heuristic function. The major work is done in this class as it decides where to shoot and how to shoot. This class consists of various different functions to evaluate the heuristic function and to decide the shot. These functions with their descriptions are given below:

   a) **getRoundStone(List**$< ABObject >$ **blocks, List**$< ABObject >$ **pigs)** - This function selects a round stone from all the blocks and returns a maximum are based on some conditions of choosing the round stone like maximum area round block, location of pig relative to that of the round block, etc.

   b) **getTargetPig(List**$< ABObject >$ **blocks, List**$< ABObject >$ **pigs)** - This method calculates the target pig with the help of a heuristic function defined in the term paper. It calls generateHeuristic() function within itself.

   c) **getLowerBlock(ABObject resBlock,List**$< ABObject >$ **blocks)** - This method finds the block on which the target pig(resBlock) lies. This is required according to the strategy in order to get the appropriate block to be shot at.

   d) **getTargetBlock(ABObject lBlock,List**$< ABObject >$ **blocks)** - This method finds the block that touches the block on which pig lies(lBlock) and which lies on the left of the pig.

   e) **getTargetBlock1(ABObject resBlock,List**$< ABObject >$ **blocks)** - This method is called to select a block if the lower block of the pig(lBlock) is null i.e. the pig is resting on the ground. Here, the block is selected based on minimum distance from the pig and the one which is to the left of the pig.

   f) **generateHeuristic(List¡ABObject¿ pigs,List**$< ABObject >$ **blocks)** - This function calculates the value of the heuristic function for each pig and stores it in an arrayList.

   g) **solve()** - This is the main method of Heuristic.java. First of all, it calls getRoundStone() and sets the target stone to the stone returned by getRoundStone(). If getRoundStone() returns null, it calls getTargetPig() method. It then calls getLowerBlock() to find the block on which target pig lies. Finally, it calls getTargetBlock() or getTargetBlock1() to get the target block. If this target block is not null, the same block is returned. Otherwise, pig becomes the target block and is returned.

   h) **distance(Point p1, Point p2)** - This function takes two points and calculates the distance between them. This is helpful in calculating distance between two blocks, bird and pig, pig and pig etc.

   i) **blockOrient(ABObject b)** - This function return true if the orientation of the block is vertical else it returns false.