**Design Document for Snowball Fight Bot**

Team members: Annabel Qi, Kevin Huang, Kevin Kim

**Strategy Descriptions**

**Strategy 1 (TALBOT)**

The first strategy relies on the lead in score in late game instead of focusing on the early games. Its goal is to reach the 30 round limits with a lead in score. This strategy has two major parts in it, the first part is a function that guesses the opponent’s move based on the situation the opponent is in, then pick the counter move to it. The second part is to let the bot not to waste any resources like snowballs and ducks.

The Guessing Function

This guessing function guesses the opponent’s move based on the situation they are in. It uses the opponent’s move based on the amount of snowballs both players have, the number of ducks both players have, the lead in the score. For example, if the opponent has 3 or more snowballs, and if our bot has less snowballs than him and both players have ducks. If the opponent is leading in score, then he is most likely to throw, so in that case, our bot would choose to duck. However, if both players ran out of ducks, and the opponent is leading in score. Then the opponent is likely to throw, but in this case, our bot would choose to throw to cancel his throw.

The getMove Function

This function tells our bot not to waste our resources if we are approaching the round limit of 30 rounds. For example, if we are leading in score, and the amount of snowballs and the number of ducks we have combined is greater than the round left. Then our bot would not choose to reload and use all our resources to extend the game to 30 rounds and win by the lead. However, if the opponent is leading in score, but we still have enough resources to the end of the game, our bot would still only choose between duck and throw based on our guesses from the guessing function.

The Opening Move

Since our bot wants to survive as long as possible and beat the opponent in late game, therefore our bot will always choose to reload on the first round, because that would give us a lead in number of snowballs even though we might get hit if the opponent chooses to throw.

**Strategy 2 (Probability)**

Our second strategy relies heavily on adjusting the probabilities of the three moves to obtain maximum success. There are no set choices, and the moves are affected by chance and random choice.

The program starts with a set value for the THROW probabilit, RELOAD probability, and DUCK probability. The reload probability is higher than the throw and duck probability, because it is the safest method. If we throw and the opponent reloads, they will have two snowballs while we will have zero, putting us at a great disadvantage for the later steps in the game. In addition, there is the chance that the opponent will throw or duck, and then we would be wasting a snowball.

After the starting move, the program increases or decreases the chances of us throwing, ducking, or reloading according to different situations. Our situations are separated by the number of the opponent’s snowballs, opponent’s ducks, and opponent’s reloads. Each of these situations then expand into more specific cases. For example, if the opponent has more than 7 snowballs, they will be more likely to throw next since they have an abundant amount of snowballs. To combat this, the program will increase our chances of ducking. We also have different situations by the number of the our own snowballs, ducks, and reloads. For example, if we only have one snowball left, the program increases our chances of reloading. As the program runs through the different situations, it recalculates the probability of us throwing, ducking, and reloading.

In addition, the program prevents us from making obvious mistakes. If the opponent has no snowballs left, we will not duck. If our number of snowballs is greater than their number of snowballs and their ducks left combined, then we continuously throw because they cannot defeat us by throwing and ducking anymore. If the opponent’s score is higher, then we increase our probability to duck since we are in a more vulnerable position.

Lastly, the program prevents cheating. The program checks if we have reached 10 snowballs, if we have none, and if we have reached our maximum ducks.

**Team Members’ Roles and Responsibilities**

**Kevin Huang:** Came up with the idea of not wasting any resources and to extend the round to 30 and win by a lead in the score. Created the guessing function and the getMove function in the first strategy. Fixed several bugs in the second strategy.

**Kevin Kim:** Came up with different ideas of approaching each strategy, provided advice for the logical aspect of the program.

**Annabel Qi:** Came up with and coded the second strategy using probability. Helped fix errors in both programs.

**Test Results**

From testing, our first strategy (TALBOT) was the strongest program. It defeated the probability strategy 989 times out of 1000.

|  |  |  |
| --- | --- | --- |
| Bot Testing Against | First Strategy (TALBOT)  Win, Loss, Tie Rate (%) | Second Strategy(Probability)  Win, Loss, Tie Rate (%) |
| RandomRoy | 89.8, 9.8, 0.4 | 72.2, 27.7, 0.1 |
| DecentAIPlayer | 80.2, 6.6, 13.2 | 61.6, 38.4, 0 |
| HalTheHoarder | 54.5, 45.5, 0 | 59.6, 40.4, 0 |
| NervousNick | 100, 0, 0 | 30.8, 69.2, 0 |
| SlightlySmartSue | 76.5, 21.8, 1.7 | 65.7, 34.2, 0.1 |
| Second (other) Strategy | 98.9, 0.9, 0.2 | 1.1, 99.1, 99.8 |

The reason why that our first strategy only has 54.5% win rate against HalTheHoarder because it save up 4 snowballs, then throw continuously, and our guessing function only guesses the opponent’s move in the current situation, but does not recognize patterns in the opponent’s move. So when HalTheHoarder uses a pattern of saving snowballs then throw, our guessing function would guess the incorrect move.

The second strategy has a very low win rate against NervousNick because the second strategy heavily relies on detecting the patterns of the opponent’s moves. For example, if the opponent has reloaded two times, we suspect a higher chance of them throwing in the next move, therefore we are more likely to duck. However, NervousNick only throws and reloads, resulting in it having no real pattern or strategy. Therefore the program’s usage of calculating probability chances according to opponent’s numbers of ducks and snowballs are never put into use.