Hungry Birds

Aman Mendiratta, Gaurav Gupta, Meenakshi Yellepeddi, Sugam, Sandeep Kaur
School of Computer Science
University of Windsor, Ontario, Canada

***Abstract*—In present scenario with great technological advancements in every aspect of life, people are very much depending on technology for building logical skills and leisure activities. To make it logically functional and also fun activity, we are making a mind exercise game for audience above twelve years.**

We have created a standalone board game known as Hungry Birds game which has to be played between two players- a larva and group of four hungry birds and This can be played in two ways either human vs human or human vs artificial intelligence. To play this game players need to calculate the next best move so that they can win the game where each mode either bird or larva has different specification to win the game. By trying to guess or calculate the next best move, we are trying to make the game a strategy game which needs players autonomous decision making and logical skills.

Keywords*—*Artificial Intelligence, MinMax algorith, Heuristic Approach, LaTeX, RAD.

I. Introduction

The use of Artificial Intelligence is growing day by day as its application are making the most of the day to day activities very easy. We are using one of the Artificial Algorithm MINMAX to build a board game. As our game is a two player game. Hence, we use this algorithm. To implement the functionality and algorithm, we use python because of its flexible nature.

For the calculation of move of larva and bird we made heuristic function and it has written in such a way so that AI can outsmart the human moves . If human is playing as larva and AI as bird then birds has to move in such a fashion so that they can block the path of larva.

The motivation to build this project came from key values like strategy, patience and concentration. We wanted the users be able to users their analytical and logical skills to solve this game so that it can improve their abilities but also at the same time have fun playing the game. Hence, our team came up with the idea to develop this combination of fun and strategy game. We also want to make this game simple so that users can understand the game while there are trying to play.

II. Project Details and Methodology

A. Process Model used

We have used Rapid Application Development model in this project. Rapid application development is a software development methodology that uses minimal planning in favour of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product. In this model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed pre-planning required , it makes easier for us to incorporate the changes during the development process.

Since its a game which needs features to be added which work and removed which won't work at different phases we choose RAD model and also our heuristic function required us to change our game frequently so a prototype would be a better choice to make clients or customers understand our game.

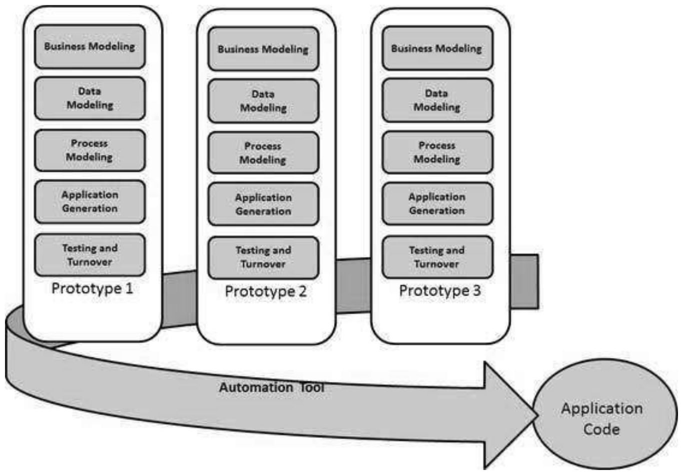


Fig. 1: RAD model

B. Project Plan

We started by splitting the work according to the skill-set of the members. We decided time frames for each task and tried to finish them accordingly by forming cross-functional teams.

We kept record of the progress in the form of project logs. Code reviews were conducted at regular intervals, and modifications were made simultaneously.

III. Requirements

A. Hardware Requirements

- Hard Disk: 20GB and Above

- RAM: 512MB and Above
- Processor: Pentium V and Above
- Internet Connection

B. Software Requirements

- Windows Operating System 7 and above
- PyCharm

C. Functional Requirements

- Enter the game
- Choose with whom to compete with either Human or Machine
- Choose the option: Select you want to play either as a Bird or Larva
- Play your moves: Move the larva (by protecting it from the bird)
- Strategies your moves: Move the Bird (by blocking larvas moves)

D. Technologies Used

- Programming Language :: Python (object oriented high level programming language used for RAD application)
- IDE :: PyCharm
- Testing :: Manual Testing (performed by team member)
- Version Control :: Git
- Project Management Software :: Redmine

IV. Architecture

Hungry birds is an adversary game between 2 players: a larva, and group of 4 hungry birds. The game is played on an 8x8 board that represents a yard. Yard positions are numbered horizontally from A to H and vertically from 1 to 8. 4 birds are located at any random positions.

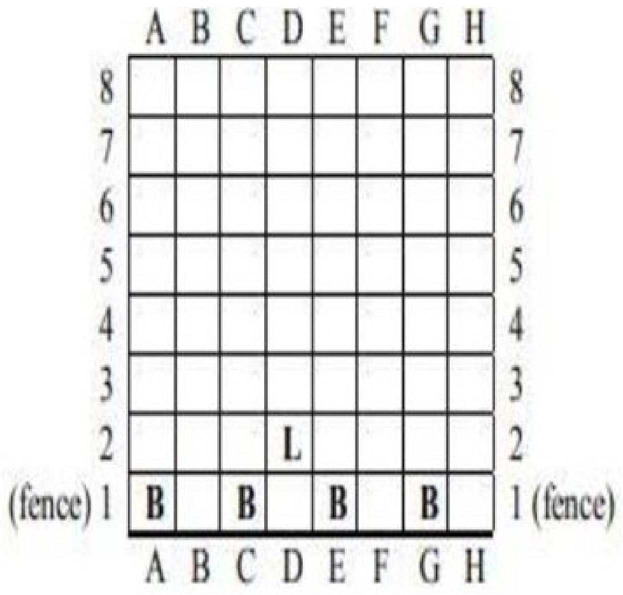


Fig. 2: Initial layout of board

Beyond line 1, there is a fence that birds are not able to cross, but the larva can. The goal of the larva is to reach line 1, before the birds catch it. To do that, there are some rules for each of the player so that we can make it more interesting. And also to make the project technologically more advanced we will develop a basic Artificial Intelligence Player and one Regular (Human) Player.

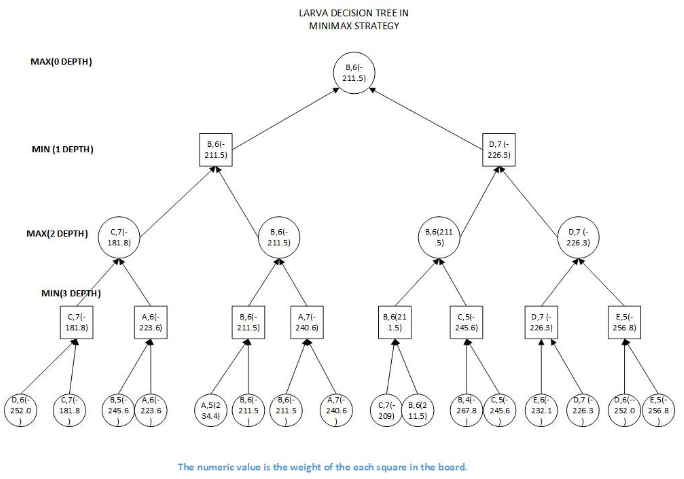


Fig. 3: Decision tree for Larva moves

V. Implementation and Code design

After numerous tries to create a heuristic which is smart and also fast to outsmart the human moves we decide to keep it simple but at the same time make it feature driven so that each we add the feature in form of formula we can make the AI smarter. The A.I. uses the Minimax algorithm, even though alpha-beta pruning is implemented but commented out as it made no big difference. In order to respect the limit of 3 seconds per move, the depth of the game tree is fixed at 3.

The heuristic calculates the sum of weight x value of birds or larva with a penalizing factor.

$$f(H) = (a1-cp)+(a2+hp2)+(a3+hp3)+(a4+hp4)+(a5+hp5)-(sumofmanhattandistancesoflarvafrombird)$$

where cp is column penalizing factor and hp(i) is penalizing factor depending on if bird is above larva.

The idea was to find these penalizing factor's optimum values so that the move made or decided by the AI was intelligent and was following the basic rules of the game. After continuous testing with different values for penalizing factor, the most optimum values I have arrived at is

For Columns it was :
'A'=1.5,
'B'=1.4,
'C'=1.3,

'D'=1.2,
'E'=1.2,
'F'=1.3,
'G'=1..4,
'H'=1.5
and for height it was 1.5 .

This function is defined for a1, a2, a3, a4, a5 = 1,-1,-1,-1,-1, the penalizing factor with the value assigned compared to other penalizing factor value tested was found more better, since the AI would make moves accordingly. The heuristic function developed is a feature driven heuristic, mainly the features include the larva needs to take birds up to win the game and as for birds they try to create a wall or drive the larva into a corner to win the game. After some observations, few more features could be implemented to make the AI better but Ive stuck to the simplest form for this prototype now. In terms of the search algorithm, Ive implemented both minimax and alpha beta pruning , and found very minute differences between them. Since it didnt help much with the decision making for the heuristic values Ive commented out alpha beta pruning in the program and used minimax algorithm. Since its python scripting language implementing a depth 4 or depth 5 game tree made the move take a little longer than 3 seconds.

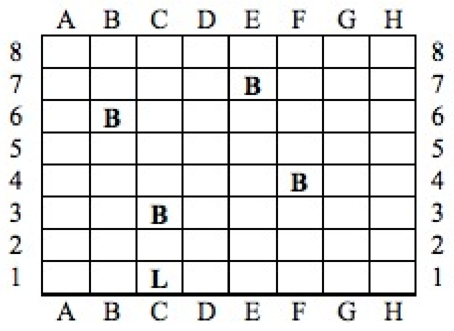


Fig. 4: Win for the Larva

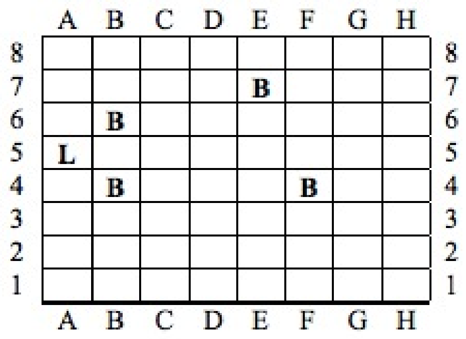


Fig. 5: Win for the bird

VI. Testing

We identified the challenges and risks in our application, and then defined the test scenarios.

An overview of the testing that we performed are as follows:

A. Unit Testing

Unit testing was done while we were developing the independent components of the application. The moves of all the birds or larva along with their positions are independently tested so that they have appropriate time and distance in moving.

B. Black-box and White-box Testing

In this we tested the whole internal structure of the program. We performed this by making every line of code in the program to execute at least once. In this, the application is tested at the unit and system levels. We also perform testing for all the possible inputs and compared the actual output with the expected output.

C. System Testing

System testing was performed at the end to check if the whole game was able to work properly with less execution time and good service. To achieve this, we checked our game by making every possible move as test case.

Also, we had to keep the depth level to minimum of 3, when we tested it for more than depth 3 the game started becoming slow, for that we need to make changes to other features.

VII. Challenges

- User Interface and experience -To design a system which is understandable as well as attractive to demand user attention.
- Scalability - To design a system which is able to handle growing amount of users and capable of maintaining the response time.
- Performance- To design the game which performs without any lags and also be able to work in any environment like mobile & web.
- Knowledge of platform - For now the game is designed to be a stand alone game built on system. We would and can make more improvements to make it work on other platforms.
- Selection of Heuristic Function took a lot of time as Movement of Birds & Larva depend upon this function. With bad selection of Heuristic function Artificial Intelligence was not working fine.

VIII. Conclusion

We were able to implement a system that could help us win the game by selecting any of two possible modes: bird or larva. Using appropriate moves, we can win the game.

Our system is not case sensitive yet .Our system is not able to handle growing amount of users and capable of maintaining the response time. For now the game is designed to be a stand alone game built on system. We would and can make more improvements to make it work on other platforms.

Therefore, the game could be used by students and people above age of 12 for sharpening their minds and enhancing their logical abilities.

IX. Future Scope

- Make it compatible on different platforms such as desktops, laptops , mobile devices, etc.
- Create an online platform to allow any guest user to play the game as free trials.
- Make interactive GUI
- Improving the security of our game application.
- Make more versions of game with better and improved AI.

Appendix A
Program Description

Program contains 3 main classes and game function method which are important.

A. *if __name__ == '__main__'*

This function is the entry point of the application. It contains a series of instructions that compose the games initialization. In class HungryBoard, it represents the board on which the game is being played.

All the values of the board are initialized to Zero values except the players and maybe their move options.

The functions and their nature are described below:

- init_board - This function initializes the players in their proper positions.
- get_board_data - This function returns the main OrderedDict variable containing all the board data as well as its size.
- print_board - This function prints the board.
- print_weights - This functions prints the weights of the board.
- lmoves - This function returns all the valid moves for larva(without checking for validity)
- bmoves - This function returns all the valid moves for a particular bird(without checking for validity)
- get_coords - Used to prompt the human player to enter coordinates.
- manhattan_distance - Calculates the manhattan distance for birds and larva.
- get_manhattan_distance_sum - Calculates the sum based on coordinates of current position of bird and larva.
- *mmh_val - It contains the informed heuristic function.
- *minimax - This method contains the minimax implementation of the game. Using but a single call to one method, the A.I. player will get its next best possible move to play.
- minimax_with_ab_pruning - This function contains the minimax implementation of the game that uses alpha beta pruning.
- parse - This method converts real world user inputs about the board to a program specific coordinate value.
- revparse - This method converts program specific coordinate value to real world user inputs about the board.

B. *Class BirdNode and Class LarvaNode*

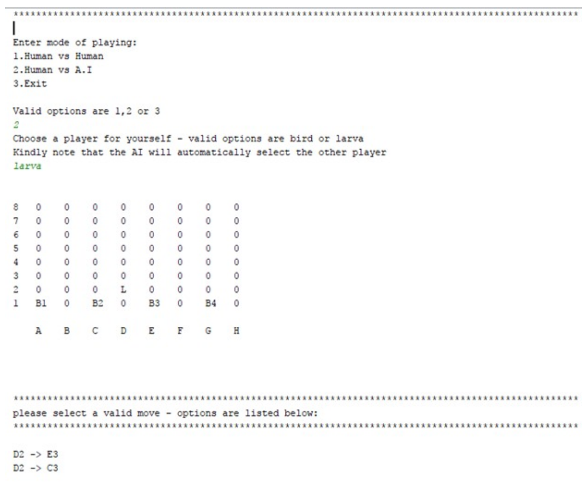
The underlying data structure that we used to create the A.I. is a tree composed of nodes. In this file, we can find the implementation of the node class. A node keeps track of a representation of the state of the game (Hungryboard), a reference to its parent and to its children.

print_tree - This function contains the implementation of the tree. It will print the tree values.

C. **Game*

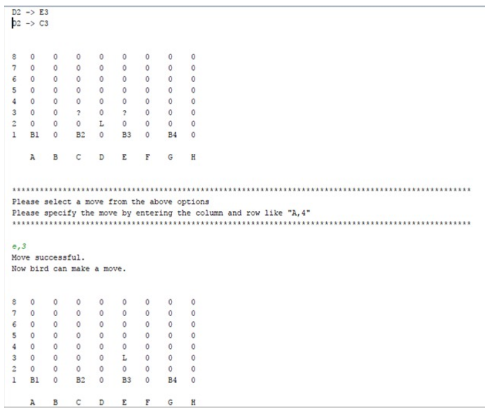
This is where the game happens (everything happens here) and It contains functions to accomplish many differently natured tasks. Tasks range from enforcing the games rules, verifying if a player won the game, printing statements for invalid moves and calculating the valid moves, evaluating the heuristic for the games current state and other functions to help the A.I and also printing the moves history.

Appendix B
Working of the game



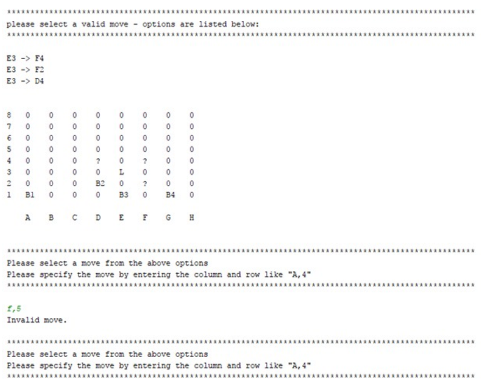
```
******************************************************************************************
|
Enter mode of playing:
1.Human vs Human
2.Human vs A.I
3.Exit

Valid options are 1,2 or 3
2
Choose a player for yourself - valid options are bird or larva
Kindly note that the AI will automatically select the other player
larva


8   0    0    0    0    0    0    0    0
7   0    0    0    0    0    0    0    0
6   0    0    0    0    0    0    0    0
5   0    0    0    0    0    0    0    0
4   0    0    0    0    0    0    0    0
3   0    0    0    0    0    0    0    0
2   0    0    0    L    0    0    0    0
1   B1   0    B2   0    B3   0    B4   0

    A    B    C    D    E    F    G    H



******************************************************************************************
please select a valid move - options are listed below:
******************************************************************************************

D2 -> E3
D2 -> C3
```

Fig. 6: Mode Selection



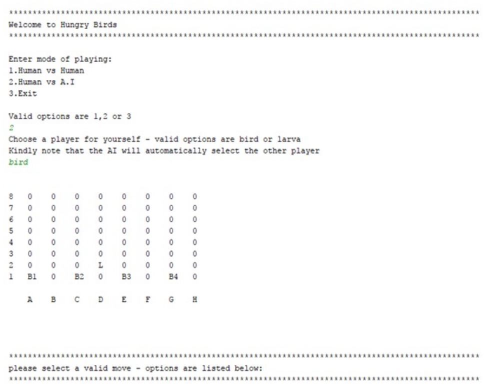
```
D2 -> E3
D2 -> C3


8  0   0   0   0   0   0   0   0
7  0   0   0   0   0   0   0   0
6  0   0   0   0   0   0   0   0
5  0   0   0   0   0   0   0   0
4  0   0   0   0   0   0   0   0
3  0   0   ?   0   ?   0   0   0
2  0   0   0   L   0   0   0   0
1  B1  0   B2  0   B3  0   B4  0

   A   B   C   D   E   F   G   H


************************************************************************************************
Please select a move from the above options
Please specify the move by entering the column and row like "A,4"
************************************************************************************************

e,3
Move successful.
Now bird can make a move.


8  0   0   0   0   0   0   0   0
7  0   0   0   0   0   0   0   0
6  0   0   0   0   0   0   0   0
5  0   0   0   0   0   0   0   0
4  0   0   0   0   0   0   0   0
3  0   0   0   0   L   0   0   0
2  0   0   0   0   0   0   0   0
1  B1  0   B2  0   B3  0   B4  0

   A   B   C   D   E   F   G   H
```

Fig. 7: Move Selection Larva



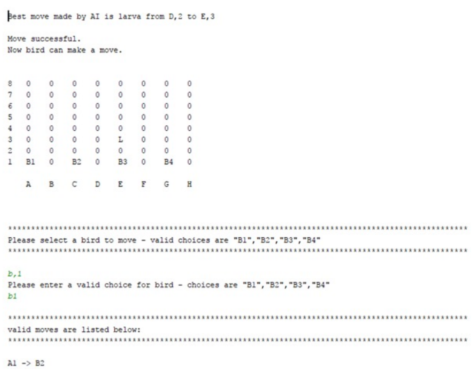
```
************************************************************************************************
please select a valid move - options are listed below:
************************************************************************************************

E3 -> F4
E3 -> F2
E3 -> D4


8  0   0   0   0   0   0   0   0
7  0   0   0   0   0   0   0   0
6  0   0   0   0   0   0   0   0
5  0   0   0   0   0   0   0   0
4  0   0   0   ?   0   ?   0   0
3  0   0   0   0   L   0   0   0
2  0   0   0   B2  0   ?   0   0
1  B1  0   0   0   B3  0   B4  0

   A   B   C   D   E   F   G   H


************************************************************************************************
Please select a move from the above options
Please specify the move by entering the column and row like "A,4"
************************************************************************************************

f,5
Invalid move.

************************************************************************************************
Please select a move from the above options
Please specify the move by entering the column and row like "A,4"
************************************************************************************************
```

Fig. 8: Invalid/Valid Move



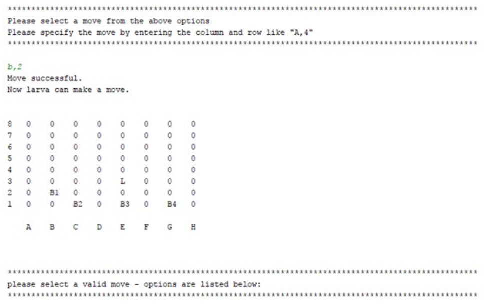
```
************************************************************************************************
Welcome to Hungry Birds
************************************************************************************************

Enter mode of playing:
1.Human vs Human
2.Human vs A.I
3.Exit

Valid options are 1,2 or 3
2
Choose a player for yourself - valid options are bird or larva
Kindly note that the AI will automatically select the other player
bird


8  0   0   0   0   0   0   0   0
7  0   0   0   0   0   0   0   0
6  0   0   0   0   0   0   0   0
5  0   0   0   0   0   0   0   0
4  0   0   0   0   0   0   0   0
3  0   0   0   0   0   0   0   0
2  0   0   0   L   0   0   0   0
1  B1  0   B2  0   B3  0   B4  0

   A   B   C   D   E   F   G   H




************************************************************************************************
please select a valid move - options are listed below:
************************************************************************************************
```

Fig. 9: Player selected Birds



```
Best move made by AI is larva from D,2 to E,3

Move successful.
Now bird can make a move.


8  0   0   0   0   0   0   0   0
7  0   0   0   0   0   0   0   0
6  0   0   0   0   0   0   0   0
5  0   0   0   0   0   0   0   0
4  0   0   0   0   0   0   0   0
3  0   0   0   0   L   0   0   0
2  0   0   0   0   0   0   0   0
1  B1  0   B2  0   B3  0   B4  0

   A   B   C   D   E   F   G   H


************************************************************************************************
Please select a bird to move - valid choices are "B1","B2","B3","B4"
************************************************************************************************

b,1
Please enter a valid choice for bird - choices are "B1","B2","B3","B4"
b1

************************************************************************************************
valid moves are listed below:
************************************************************************************************

A1 -> B2
```

Fig. 10: Bird B1 Selection



```
************************************************************************************************
Please select a move from the above options
Please specify the move by entering the column and row like "A,4"
************************************************************************************************

b,2
Move successful.
Now larva can make a move.


8  0   0   0   0   0   0   0   0
7  0   0   0   0   0   0   0   0
6  0   0   0   0   0   0   0   0
5  0   0   0   0   0   0   0   0
4  0   0   0   0   0   0   0   0
3  0   0   0   0   L   0   0   0
2  0   B1  0   0   0   0   0   0
1  0   0   B2  0   B3  0   B4  0

   A   B   C   D   E   F   G   H


************************************************************************************************
please select a valid move - options are listed below:
************************************************************************************************
```

Fig. 11: Move Selection Bird

References

[1] Fig 1: https://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm

[2] MINMAX Algorithm: http://ieeexplore.ieee.org/abstract/document/6261563/

[3] https://docs.python.org/2/

[4] http://www.tutorialspoint.com/python/

[5] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.