Project Report

Group 1
COMP2021 Object-Oriented Programming(Fall 2018)
Gao Jiacheng 17083488d
Liu Hanze 17083854d
Lian Xiang 17082482d
Xu Dunyuan 17083297d

1. Introduction

Our game is able to simulate how players play the jungle game, here is our game‘s architecture.
In total, we have a unit class which refers to the board and a animal class which refers to the pieces. In the animal class, we have four class extends the animal class which cover four different animals’ move methods.
In addition, we have cellType class and animalInfo class which contain the units’ landform information in the board and the pieces’ animal type information respectively.
Moreover, we have board class for users to know where their animals are and let users play this game.
About the design choice of the inheritance and polymorphism, we focus on animal pieces in this game. First, we consider all elements in this program: pieces and units on the board. We found that units which can show the information is enough to realize the game. And different units are only different in information. Therefore, unit class is only to contain information, no need to inherit. However, animal has different move rule, different animal needs different move method. Therefore, we make 4 classes inherit from animal class and make move method polymorphism.
We treat each piece of animal and each unit of this board as an object. Link these two kinds of objects with the fromPosition field, the same fromPosition means that they are at the same position then we can deliver the move method in Board class to the move method in Animal class. In this way, we create 16 objects storing the pieces and 63 objects storing the units on board, which makes the whole program have a higher scalability and reusability.

2. The Jungle Game

The Jungle game board represents a jungle terrain with dens, traps around dens, and rivers.Each player controls eight game pieces representing different animals of various rank following some rules. The player who is first to maneuver any one of their pieces into the opponent's den or capture all the opponent's pieces wins the game.
Our game is for players to play the jungle game on the computer. Players can start a new game or open a saved game and move the pieces in order to occupy opposite den or eat up opposite animal pieces. Specifically, when players move pieces, they must follow some rules:

1. Higher ranking pieces can capture all pieces of identical or lower ranking. (elephant > lion > tiger > leopard > wolf > dog > cat > rat).
2. However, The rat may capture the elephant, while the elephant may not capture the rat.
3. During their turn, a player must move.

4. Each piece moves one square horizontally or vertically (not diagonally).
5. A piece may not move to its own den.
6. The rat is the only animal that is allowed to go onto a water square. The rat may not capture the elephant or another rat on land directly from a water square. Similarly, a rat on land may not attack a rat in the water. The rat may attack the opponent rat if both pieces are in the water or on the land.
7. The lion and tiger pieces may jump over a river by moving horizontally or vertically. They move from a square on one edge of the river to the next non-water square on the other side. Such a move is not allowed if there is a rat (whether friendly or enemy) on any of the intervening water squares. The lion and tiger are allowed to capture enemy pieces by such jumping moves.
8. Animals capture the opponent pieces by "eating" them. A piece can capture any enemy piece which has the same or lower rank, with the following exceptions: A rat may capture an elephant (but not from a water square); A piece may capture any enemy piece in one of the player's trap squares regardless of rank

2.1. Requirements

Req01 When the program is launched, a user should be able to choose between starting a new game and opening a saved game
Software elements: in class controller, main method, use the Scanner to get user input.

Req02 At the beginning of a new game, the two players X and Y should be prompted to input their names. Then the initial board should be printed and player X should be prompted to input a command.
Software elements:
1.names of teams: In class Board, a String array called "name" of size 2 (String[2]) to store the names of both teams.
2.the whole jungle chess board is a 2D-array. When printing it, we read each grid's information and translate it into different Strings, then print out line by line.
3.a simple print statement to ask for a command.

Req03 command (open/save/move)
Software elements:
1.save: for corresponding class which need to be saved, implements Serializable, then we use the objectOutputStream to save the whole board, the animals on the board and the names of two teams.(use the writeObject method)
2.open: similar as open command, implements Serializable for all corresponding class, but this time we use ObjectInputStream and the method readObject. Read the object from the file one by one and assign it to the board, animal and name (arrays in class Board).
3.move: get the user input as a string, using scanner. Then judge whether it is legal in the Controller. Judge whether it is legal is according to its length and its range for the

two indexes. Furthermore, if it is a legal input, then go into deeper judge for the concrete position. If it is illegal, print the error message and ask the user to input another command using a while loop, whose exit condition is the win condition. Once getting the legal input, the move will test if there is a piece on that position. If not, it is still considered as illegal. For the fromposition, if it make sense, will be delivered to the move function of the animal on that position.

Req04 the text legal function (which means the chosen position is on of the units on board) will be delivered to the animal on that position. The animal on it will dynamic binding to the specific animal and use the specific move rules accordingly. If the move is judged illegal, the thisTurn field will not be changed and in the while loop, the program will print the previous board, print error messages and ask for a new move command for the same team user. If the command is valid, the unit information and the animal information will changed accordingly. This time the system will also ask for a new input. it will not exit until one team attend the win condition or one user would like to save the game.
Software elements: class: Controller, Board, Unit, Animal
method: getIfHasAnimal(), move(fromPosition, toPosition) (this move is in Board class), move(fromPosition, toPosition) (this move is in Animal class), printBoard()

Req05
After a valid move, to print the board, we create a method printBoard() which can read all elements in the two-dimension array Board[7][9]. If the unit we are reading has not animal on it, program will print out the unit's landform. If it has animal, we will get what kind of animal it is by searching the unit's position in the array animals[16] and print out the animal type.
To make that after each valid move, change a player to input the order and exit if one player achieve the goal, we define two team as 1 and -1, store the integer refer to team in the value thisTurnTeam and write a method ifChangeTeam(). There are two if condition in the method, one is check whether any den is occupied by animal: if true, change thisTurnTeam to 0 which is not belong to any of two team; the other is check whether has need to change player: the method read the static value needDoMoveAgain that is set in the every time move. If need do move again, change thisTurnTeam to negative thisTurnTeam. Then we put the move() method and ifChangeThisTurn() in a while loop. When one team's animals are all dead or any one den is occupied by the other team's animal which means to thisTurnTeam = 0, jump out of the while loop and print out this turn player's name and exit the program. How we know whether one team's animals are all dead? Using a method ifAllDead() reading all elements in the array animals[16] to see whether there is an animal's fromPosition is "Z1"(in the move method, we change the animals' position to "Z1" after they are captured).
Software elements: printBoard(), ifChangeThisTurn(), ifAllDead()

Req06
For the invalid move command, out of the move() method, we check whether the player input a valid String, like A1 is valid and RE/1E/34 are invalid, if not valid we will ask player to input valid one and print the board; in the move() method, we have considered every invalid case and print the error message and unchanged board in the

move() method. And as Req05 said, we use ifChangeThisTurn() to control whether we change player.
For invalid input in open command, a ClassNotFoundException or EOFException will be thrown.
in save command, the input is always valid, if there is not such a file, the file will be created.
Software elements: printBoard(), ifChangeThisTurn(), Animal class' move(), save(), Open(), try-catch statement.

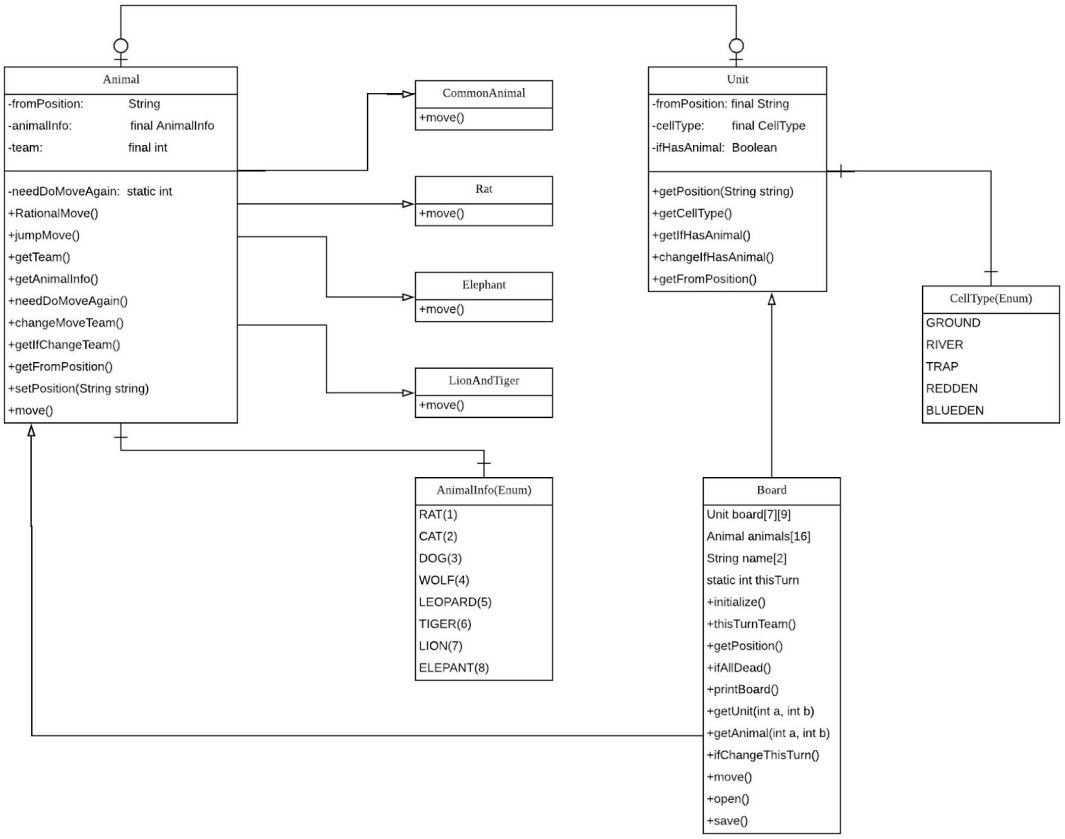
Bonus

bonus1

Using javafx to implement GUI. The whole GUI are made up by 2 main scenes, the first scene contains 3 choices('start', 'open' and 'quit'), and the second scene is the jungle chess board. And in the second scene, we have two stages to ask for user's input, the first is used for inputting the two players' names when starting a new game and the second is used for inputting the filename when saving the game. And there are many warning information and confirm information dialog windows, they will appear to remind the user when saving, opening, quitting, wining and invalid moving. We inherited the 'Button' class in javafx to create two new classes 'animal' and 'grid', and put each animal's picture on these instances of the 'animal' class. These two classes represent that position on the board is an animal or a empty grid. And we sense the click event to move the animals, the rule is the same as we write in implementing CMD.

2.2 Design

Model design:



2.3. Quick Start Guide

command line part:

1.When you first get into the game, you can choose to start a new game or open a saved game by typing in “n” or “o”:

```
Enter 'n' to start a new game,
Enter 'o' to open a saved game
n
```

2.Then you enter your team name for the left hand side team:

```
Please enter your team name:
A
```

Then the right hand side team:

```
Please enter the other team name:
B
```

3. In each step, you will be ask to input a command(open/save/move)
if you input is invalid, you will be ask to enter until it is valid.
4.if your input is move, you can move legally now

```
Please enter the command.(open/save/move):
move
A TEAM PLEASE!

move from: B2
move to: C2
TIG     ELE             RAT     LIO
            ~~~ ~~~ ~~~     DOG
^^^ CAT WOL ~~~ ~~~ ~~~ LEO     ^^^
[ ] ^^^                     ^^^ [ ]
^^^     LEO ~~~ ~~~ ~~~ WOL     ^^^
    DOG     ~~~ ~~~ ~~~     CAT
LIO     RAT             ELE     TIG
```

if your move is illegal, you will be ask to input a command again.

5.if your input is save, you will be asked for a path to save current game

```
Please enter the command.(open/save/move):
save
Please enter a path to save:
```

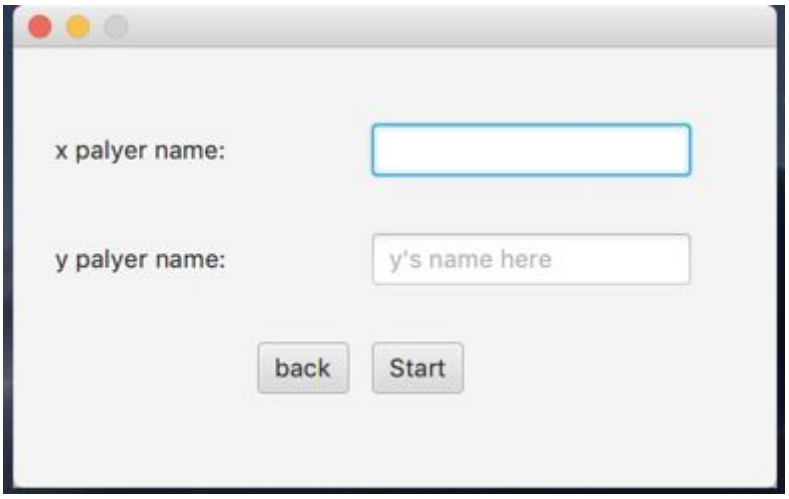
you can input an existing file path to save the game, if your input path doesn't exist, a new file will be created in that path to save the game.

6.if your input is open, if your current game has already been saved, you will be asked to input a path of an existing file to open.(Of course that file should have stored a jungle game) Otherwise, you will be asked to save your current game first before you can open a saved game. However, if your input file is not exist, an exception will be shown and you will be asked to input a command again.

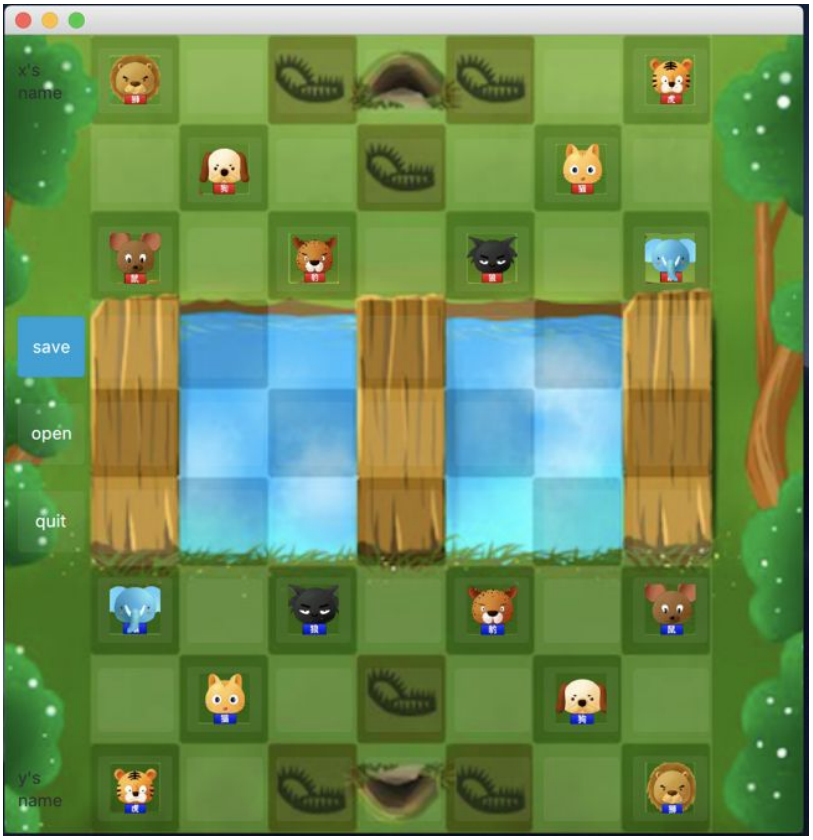
GUI part:

1.The first scene

2. If click 'start' button and then users are required to input two usernames:

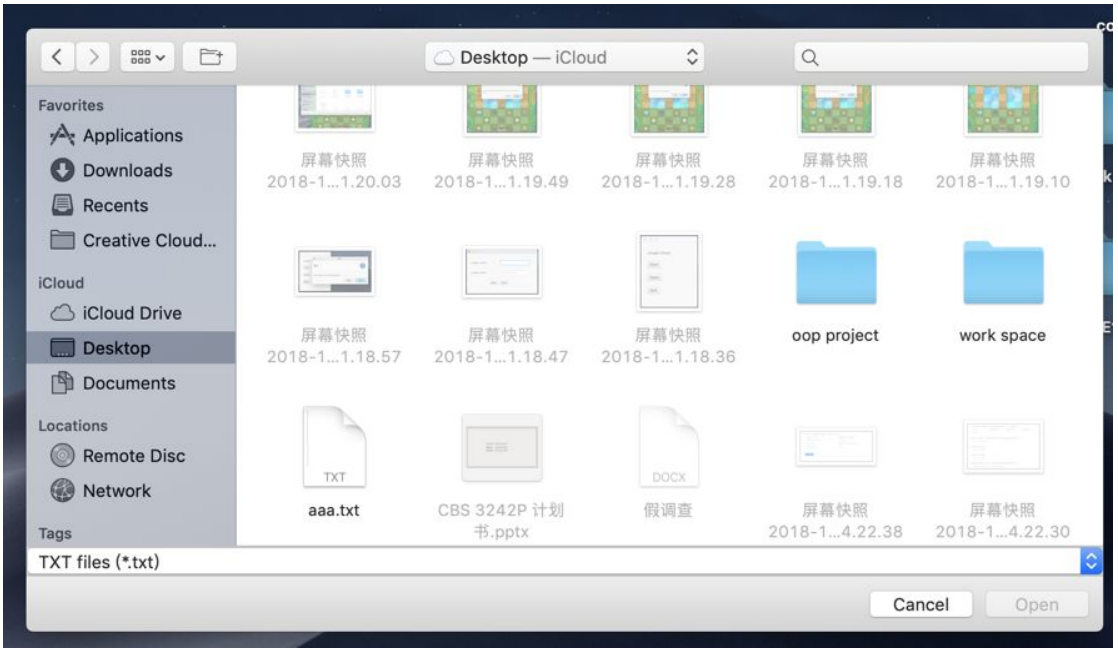


If click ‘back’ button, it will return to the first scene.
If click ‘start’ button, it will jump to the second scene (the new game scene).

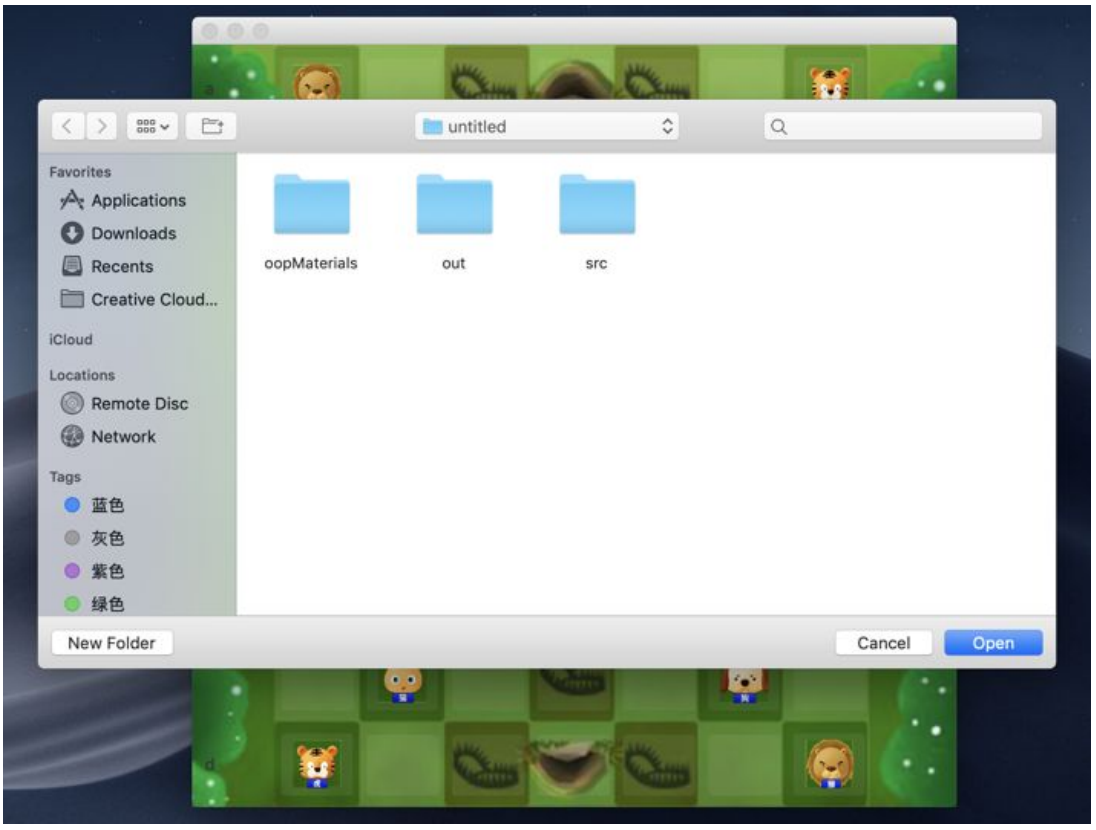


3.If click ‘Quit’ at the first scene, it will ask if users really want to quit the game, if user click ‘confirm’, the whole procedure will be exited.

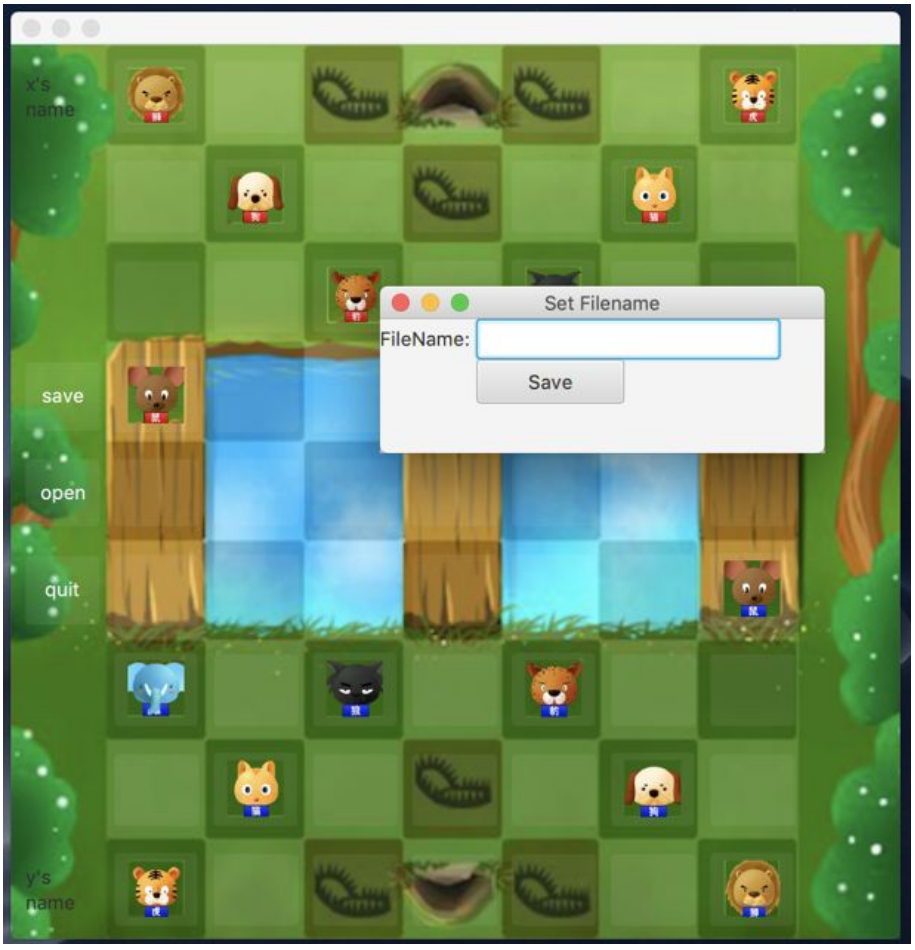
4.If click ‘Open’ at the first scene, it will open a window for users to choose which txt file to open. Only the ‘.txt’ file will be lighted, but other types files will be shown as dark. After choosing one ‘.txt’ file, users can then click ‘open’ button, and the pre-saved game will be opened.



5.If we click the ‘save’ button in the game, it will show a window to let user to select which path the game will be saved. After choosing, the users can click ‘open’ button and the path will be recorded.

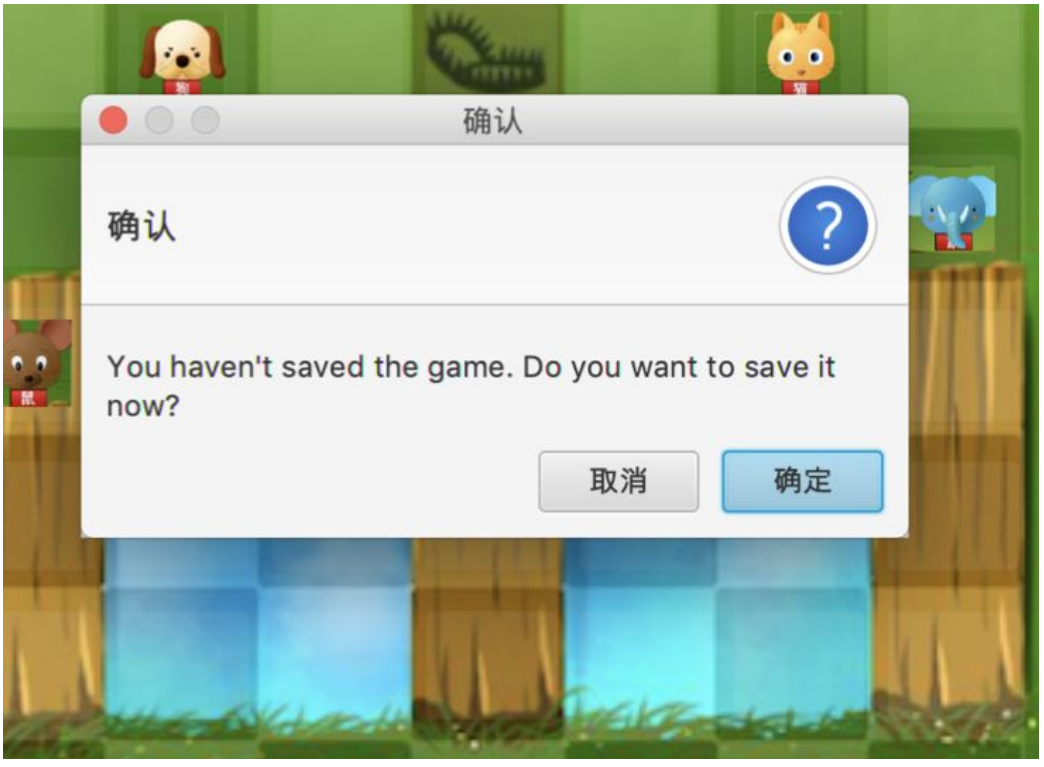


6.After choosing the path, it will ask user to insert the filename to be saved. And after clicking the ‘save’ button, the file will be saved as filename.txt at that path.



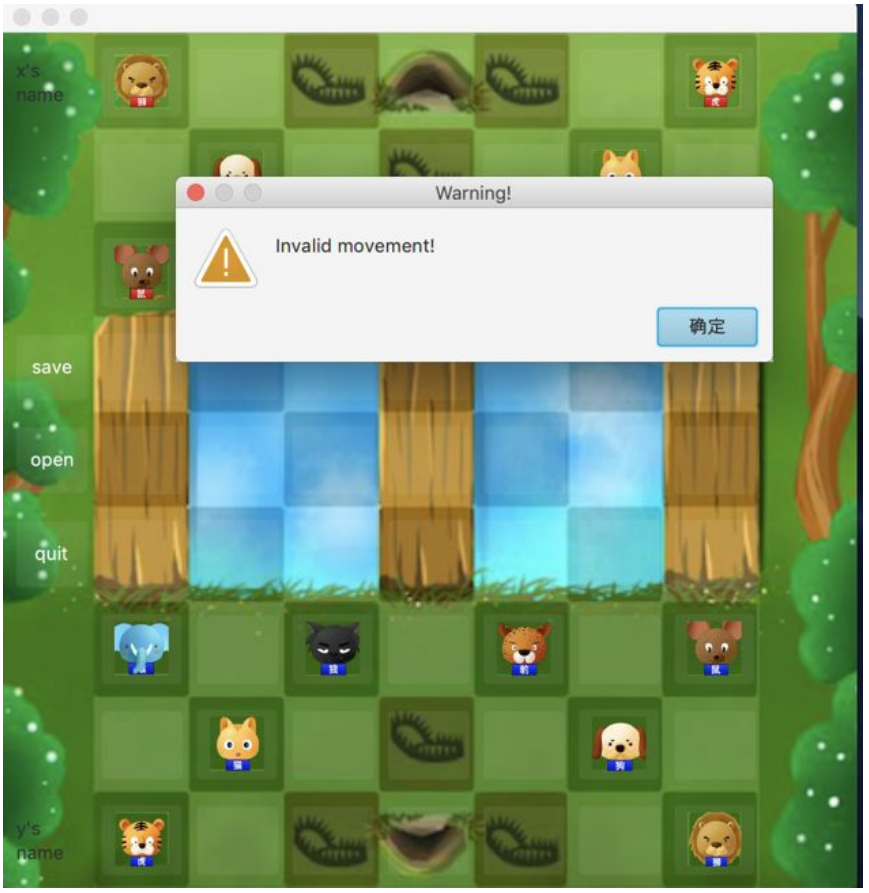
7.If we click the ‘save’ button in the game, it will show a confirm window to ask user if he really want to open a new game (same as the open operation before).

8.If user confirm to open a new game and the game is not saved now, it will ask user if he want to save this game first, if user click ‘yes’, then it will jump to the save page (same as the save operation before)



9.If we click the ‘quit’ button in the game, it will show a confirm window to ask user if he really want to quit this game. And it will also check if the game is already saved (same with the open operation above)

10. If the valid movement is appeared. The warning message will show.



11.If one side win the game, the winner name will be print out and if user click ‘confirm’ the whole program will be exited.