Australian National University

Assignment2 Group Presentation Tuesday09(a)

Presented by: Chunhao Li
Mingchao Sima
Haoyan Liu

specialties

- The button is filled with gradient.
- 4 modes, greatly friendly to users.
- Press ‘space’ to return to the last step, much more convenient to users.
- “Debug mode” provides irreplaceable opportunity for developers to improve it.

specialties

- High –level javaFX (Special picked background, Distinctive icon , Professional name , Beautiful music and Multiple modes
- 4 modes, greatly friendly to users.
- Press ’Q’ to exit
- Strong interaction with player

Code analysis

- Properly named and well-documented
- “Help methods” are set just under the main method, easy to read.
- Clear logic and simple code.
- Times of improvement and simplify.
- Perfect time and space performance

Design approach

Task8 Compute the basic score

- -countExitScore
- -countErrorScore

Design approach

Task10 generateValidMove

- -Stupid: Search all the grids and find unused ones, then move if valid.
- -Better!:
- getAvailableGrids

Method approach GetAvailableGrids

```
public static HashSet<String> getAvailableGrids(String boardString) {
    String[] exits = {"A1", "A3", "A5", "G1", "G3", "G5", "B0", "D0", "F0", "B6", "D6", "F6"};
    List<String> availableExits = new ArrayList<>();
    Collections.addAll(availableExits, exits); // all exits' positions
    List<String> placedTiles = new ArrayList<>();

    for (int i = 0; i+5 <= boardString.length(); i+=5) {
        String grid = boardString.substring(i+2, i+4);
        availableExits.removeIf(g -> g.equals(grid));   // remove exit's grid if placed
        placedTiles.add(boardString.substring(i, i+5)); // add all unused exit together
    }
    HashSet<String> availableGrids = new HashSet<>(availableExits);
    for (String tile: placedTiles) {
        List<String> adjGrids = getAdjGrids(tile);
        availableGrids.addAll(adjGrids);    // add all adjacent grids
    }
    return availableGrids;
```

Design approach
Task 12

Recursion and Backtracking

```
    private void findLongestRoadRec(String s, HashMap<String, Boolean> visited,
                                    int prev, int[] max) {
        visited.put(s, true);
        int curr = 0;

        for (String adjTile : adj.get(s)) {
            if (!visited.get(adjTile)) {
                curr = prev + 1;
                findLongestRoadRec(adjTile, visited, curr, max);
            }
            max[0] = Math.max(max[0], curr);
        }
        visited.put(s, false); // back tracking

    }
}
```

Design approach
Task13

Main idea: Counting score

1. Board score
- First half turn
- Last half turn

```
int[][] boardScore = {{10, 10, 10, 10, 10, 10, 10},
                      {10, 20, 20, 20, 20, 20, 10},
                      {10, 20, 50, 50, 50, 20, 10},
                      {10, 20, 50, 50, 50, 20, 10},
                      {10, 20, 50, 50, 50, 20, 10},
                      {10, 20, 20, 20, 20, 20, 10},
                      {10, 10, 10, 10, 10, 10, 10}};
```

Design approach Task13

2. Connected to existing route or exits

```
ArrayList<String> adjTiles = getNeighbours(aMove, placed);

if (!adjTiles.isEmpty()) {
    score += 40; // connected to existing route
} else {
    score += 10;
}
```

3. Error count

```
score += countErrorsScore(newBoardString) * 15;
```

Difficulties

- Multiple modes independently work
- Implement the rules
  - In each turn, must place all four tiles if possible
  - Special tiles constraint
    (At most 1 per turn, 3 per game, no repetition)
- Clear the game state

Future improvement

- A stronger AI can be created
- JavaFX part like multiple language, music when click
- A changeable size screen can be designed to replace the current fix-sized one.
- When dragging the tracing area is too small . The way to solve it is to draw a frame and trace the whole frame.