Escuela Colombiana de Ingeniería Julio Garavito

Programación Orientada a Objetos 2024-2

Pruebas de aceptación Proyecto Inicial

Cristian Santiago Pedraza Rodríguez

Andersson David Sánchez Méndez

13 de noviembre de 2024

This user story covers three distinct game modes, offering players the ability to manipulate and solve puzzles with various initial and final configurations.

The interactions include movements, glue application, deletions, holes, and the ability to adjust visibility and check if the goal has been reached (isGoal).

Each action is defined to respond appropriately to the board configuration and the properties of the tiles and glues.

FIRST CASE

h,w with h,w > 0 and h,w <=500, to create h,w and addTile(1,1)

We set stages where h, w are not valid

finish() //Correct

SECOND CASE

:))) Stage where ending is null

Here we set valid ending:

ending = {

{'y', 'r', 'g', 'b', 'y', 'b', 'g', 'r', 'y', 'g'},

{'*', 'b', 'g', 'b', 'r', 'g', 'b', 'y', '*', 'g'},

{'b', '*', 'y', 'r', 'y', '*', '*', 'r', 'y', 'b'},

{'r', 'g', 'b', '*', 'r', 'g', 'b', 'y', 'r', 'y'},

{'*', 'b', '*', 'r', '*', 'b', 'g', '*', 'y', 'b'},

{'g', 'r', '*', 'b', 'g', 'r', '*', '*', '*', 'r'},

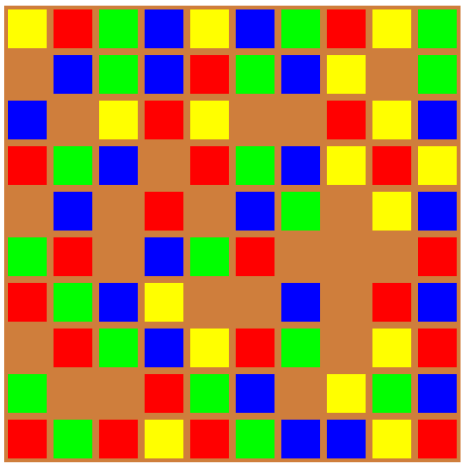
{'r', 'g', 'b', 'y', '*', '*', 'b', '*', 'r', 'b'},

{'*', 'r', 'g', 'b', 'y', 'r', 'g', '*', 'y', 'r'},

{'g', '*', '*', 'r', 'g', 'b', '*', 'y', 'g', 'b'},

{'r', 'g', 'r', 'y', 'r', 'g', 'b', 'b', 'y', 'r'}

}



Then, we set stages for various methods when the code can fail because the marathon problem requirements say it.

exchange, //Correct

fixedTiles(), //Correct

tilt('f'), //Invalid direction

tilt('r'), //Correct

addGlue(3,3), //Correct

tilt('l'), //Correct

deleteTile(3,1), //Not delete tile that is Stuck

deleteTile(8,2), //Correct

makeHole(3,2), //Only make hole in an empty tile

makeHole(8,2), //Correct

makeHole(8,2), //Already has a hole

tilt('d'), //Correct

misplacedTiles() = 73, //Correct cuz hole doesn't take account

deleteGlue(-1,4), //Invalid position

deleteGlue(3,1), //No glue to remove

relocateTile({8,2},{0,8}), //Not move hole tile

relocateTile({3,2},{0,8}), //Not move tile that has glue

deleteGlue(3,2), //Correct

isGoal()=false, //Correct

ok = false, //Correct

relocateTile({0,0},{0,8}), //Not move non-existent tile

relocateTile({1,0},{0,2}), //Totile is occupied

relocateTile({1,0},{0,8}), //Correct

tilt('l'), //Correct

addGlue(8,1), //Correct

addGlue(8,1), //Already has glue

deleteTile(8,1), //Not delete cuz it has glue

deleteTile(6,1), //Correct

deleteTile(5,1), //Correct

addGlue(7,1), //Correct

tilt('r'), //Correct

actualArrangement(), //Correct

ok = true, //Correct

finish()

This ending constructor tested different methods including cycle1, cycle2

THIRD CASE

:)) Stage where starting or ending null

Here we set valid starting and ending matrixes:

starting = {

{'y', 'r', 'g', '*', 'y', 'b', 'g', 'r', 'y', '*'},

{'*', 'b', 'g', 'b', 'r', 'g', 'b', 'y', '*', 'g'},

{'b', '*', 'y', 'r', 'y', '*', '*', 'r', 'y', 'b'},

{'r', 'g', 'b', '*', 'r', 'g', 'b', 'y', 'r', '*'},

{'*', 'b', '*', 'r', '*', 'b', 'g', '*', 'y', 'b'},

{'g', 'r', '*', 'b', 'g', 'r', '*', '*', '*', 'r'},

{'r', 'g', 'b', 'y', '*', '*', 'b', '*', 'r', 'b'},

{'*', 'r', 'g', 'b', 'y', 'r', 'g', '*', 'y', 'r'},

{'g', '*', '*', 'r', 'g', 'b', '*', 'y', 'g', 'b'},

{'r', 'g', '*', 'y', 'r', '*', 'b', '*', '*', '*'}

}

ending = {

{'y', 'r', 'g', '*', 'y', 'b', 'g', 'r', 'y', '*'},

{'*', 'b', 'g', 'b', 'r', 'g', 'b', 'y', '*', 'g'},

{'b', 'r', 'y', 'r', 'y', '*', '*', 'r', 'y', 'b'},

{'r', 'g', 'b', '*', 'r', 'g', 'b', 'y', 'r', '*'},

{'*', 'b', '*', 'r', 'y', 'b', 'g', '*', 'y', 'b'},

{'g', 'r', 'b', 'b', 'g', 'r', '*', 'g', '*', 'r'},

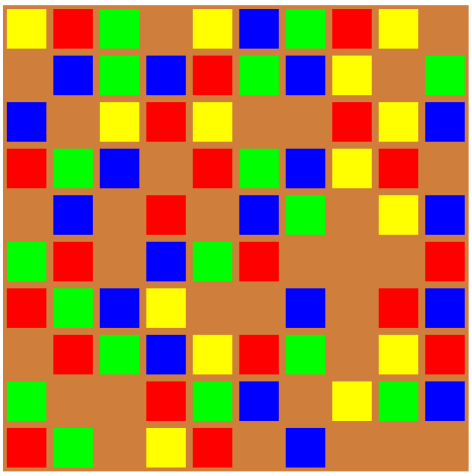
{'r', 'g', 'b', 'y', '*', '*', 'b', '*', 'r', 'b'},

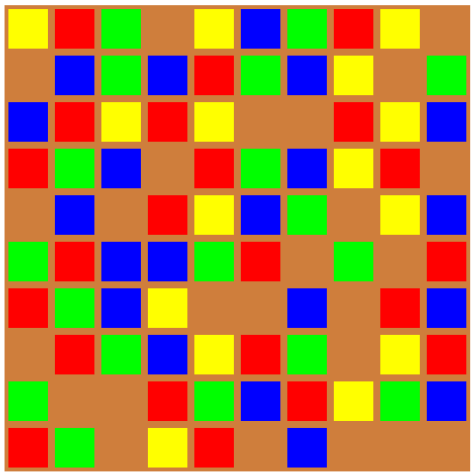
{'*', 'r', 'g', 'b', 'y', 'r', 'g', '*', 'y', 'r'},

{'g', '*', '*', 'r', 'g', 'b', 'r', 'y', 'g', 'b'},

{'r', 'g', '*', 'y', 'r', '*', 'b', '*', '*', '*'}

}





Test different types of tiles, and types of glues with different methods and to look for the good functionality of these requirements.

isGoal() = false, //Correct

fixedTiles(), //Correct

addTile(0,0), //Not addTile occupied

addTile(1,0,'f'), //Invalid label

addTile(2,1,"uh g"), //Not valid type tile

addTile(2,1,"fi r"), //Correct

addTile(4,4,"ro y"), //Correct

addTile(5,7,"fr g"), //Correct

addTile(5,2,"fl b"), //Correct

addTile(8,6,"wi r"), //Correct

addGlue(8,1), //Not existent tile

relocateTile({2,1},{89,6}), //Exceed puzzle space

relocateTile({2,1},{9,9}), //Not relocate fixed Tile

addGlue(0,6,"superFragil"), //Invalid type glue

```
addGlue(0,6,"super"), //Correct
addGlue(5,9,"fragile"), //Correct
makeInvisible(), //Correct
makeVisible(), //Correct
isGoal()= true, //Correct
actualArrangement(), //Correct
deleteTile(2,1), //Not delete fixedTile
makeHole(-5,3),  //Not negative position
deleteTile(7,2), //Correct
makeHole(7,2), //Correct
deleteTile(9,6), //Correct
tilt(), //Correct
tilt('d'), //Correct
tilt('d'), //Correct
deleteTile(9,6), //Not delete wildTile
deleteTile(9,2), //Correct
addGlue(8,7,"super"), //Not glue freelanceTile
addGlue(1,1), //Correct
deleteTile(4,9), //Correct
tilt(r'), //Correct
ok = true,  //Correct
isGoal= false, //Correct
finish()
```

Here, we tested cycle1,cycle2,cycle4.

FOURTH CASE

:)) Here we test the marathon problem Tilting tiles. We test the two methods, simulate and solve in different stages, so here is the test:

starting = {

{ '*', 'r', '*', '*', 'y' },

{ 'r', 'g', 'y', 'b', '*' },

{ '*', 'b', '*', '*', 'r' },

{ '*', 'y', 'r', '*', 'b' },

{ 'y', '*', 'b', 'g', '*' },

{ 'r', '*', '*', 'y', 'g' },

{ '*', 'g', 'y', '*', 'r' }

}

ending = {

{ 'r', 'y', 'r', 'r', '*' },

{ 'y', 'b', 'b', '*', '*' },

{ 'g', 'b', 'g', '*', '*' },

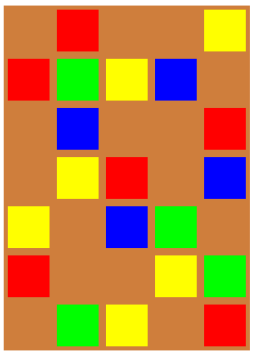
{ 'y', 'r', 'g', '*', '*' },

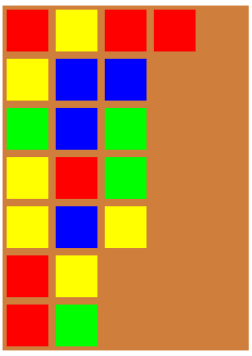
{ 'y', 'b', 'y', '*', '*' },

{ 'r', 'y', '*', '*', '*' },

{ 'r', 'g', '*', '*', '*' }

}





Here we test:

tilt() in Constructor starting and ending true //Correct

simulate(starting,ending), //Correct

solve(starting,ending) //Correct

If we change some label in ending matrix, the answer will be false, or there's not possible solution.