Coloristance: Acceptance test

Latest version: 2013-05-31
Document description: This document describes the performed acceptance tests for Coloristance, our Android application. The tests below are generally connected to specific user stories.

MapModelTest.testSetPos() - Checking the setPos()-method

Test ID: T01
Test case description:
This test investigates if we are able to set the position of the player.

Precondition:
It is necessary to have knowledge how the position is represented.

Test steps:
1. setPos to 3,2
2. Assert that the x-pos is set to 3 and the y-pos to 2

Expected results:
The expected result is that both test case 1 and 2 will be true.

Related user stories
- As a user, when looking at the map, I want to see my own position.

Motivation to test:
We needed to check that it was possible to set the players position to the correct value as intended. We also needed to know that the get-method returned the correct position.

MapModelTest.testArrayLevel1() - Test that the correct level is initialized

Test ID: T02
Test case description:
This test is used to check that each room in the map has the correct roomcode.

Precondition:
Knowledge about the roomcodes in level 1

Test steps:
1. MapModel.setMap(1);
2. Store the map in a variable
3. Several steps to assert that each room has the correct code, and that it is not empty.

Expected results:
This test should result in that all the roomcodes are correct, and should therefore return true.

Related user story:
- As a user, I want to see the room I am in, with the correct doors attached to it, as seen on the map.

MapModelTest.testCorrectArrayLength() - Checking the height and width of the mapArray

Test ID: T03
Test case description:
This test investigates the height and the width of the mapArray.

Precondition:
It is necessary to have knowledge how the map is constructed.

Test steps:
1. Get the MapModel.length and test if it is equal to 8
2. Get the MapModel[0].length and test if it is equal to 3

Expected results:
The expected result is that both test case 1 and 2 is that it will be true.

Related user stories
- As a user, I want to see how big the map is for this level.

Motivation to test:
We wanted to investigate if the size of the map had the right proportions. Initially we thought that we were supposed to run our for-loop when drawing the map from 0 =< mapArray.length but this gave us a length of the map of 8, which gave us a Nullpointer Exception when the created level only was 3*7.

MapModelTest.testCodeOfRoom() - Investigates if the room have the correct code

Test ID: T04
Test case description:
This test case first sets the position on the created map. When the position on the map is set, we test if the position on the map corresponds with the correlating roomcode.

Precondition
The tester needs to know how to set the position of the player and then have knowledge of the correct roomcode that each room has.

Test steps

1. Call the method MapModel.setPos(1,1)
2. Compare the MapModel.getRoom() which contains the roomcode with "13027"

Expected results

The Expected result of this test is that the roomcode will correspond with the given string. It should result in true.

Related user story

- As a user, I want the color of the room to be the same on the middle screen as my position on the map.

Motivation:

With this test we wanted to investigate if the setPos(x,y) changed the color of the room to match the current position on the map.

MapModelTest.testMovementsOnMap() - Move on the map and test unallowed movements

Test ID: T05

Test case description:

This test investigates if the player can move on the map, which position the player stands on. It also investigates the critical movements in the game, for example moving into a black=empty room or trying to move out of bounds.

Precondition:

In order to write this test you need to understand how the each level is constructed and how you are allowed to move. It is also necessary to have knowledge about how the player position is represented.

Test steps:

Variation1:

1. MapModel.moveRight()
2. MapModel.moveUp()
3. MapModel.moveDown()
4. MapModel.moveLeft()

Variation2:

1. MapModel.moveRight()
2. MapModel.moveRight()
3. MapModel.moveUp()
4. MapModel.moveUp()
5. MapModel.moveDown()
6. MapModel.moveDown()
7. MapModel.moveDown()

After each movement, we check that the position is correct according to the precondition.

Expected results:
Variation1:

- The player position when the allowed commands above are run the player should stand on the position (2,0).

Variation2:

- The player should not be able to move out of bounds, move into a black=empty room, and therefore the player should stand on the position (1,2) when the commands are executed.

Related user story:

- As a user, when looking at the map, I want to see my own position.
- As a user I want to change room by clicking on a door.
- As a user I want to be able to move between rooms, in order to reach the end.

Motivation:
This test was performed in order to understand why the position on the map did not correlate with the player position. The second part helped us to investigate if you were able to move out of bounds.

RectModelTest.testSetRectColor() - Check if it is possible to set a color and it is stored correctly

Test ID: T06
Test case description:
This test investigates if it is possible to set a color of a room and if a color is set is it stored in the right format, and in the right variable.

Precondition:
Knowledge about the formats used for the colors in the program, and which variable it is attached to.

Test steps:

1. setRectColor("2")
2. Test if rectColor, the variable, and getRectColor() is equal

Expected results:

- When the setRectColor("") is called, it should store its value in the variable rectColor and when the getRectColor() is called it should return this value.

Motivation:
This test helped us to investigate if we were able to manually set the color of the room.

RectModelTest.testMoveColor() – Test if the color correlates with the players movement

Test ID: T07
Test case description:
This test if the color of the room is set to the correct color every player moves.

Precondition:
Knowledge about the formats used for the colors in the program, and which variable it is attached to. It is also needed to have knowledge about that the colors need to be set after every movement manually in the test, dependent on that we have separated the player and the map.

Test steps:

1. MapModel.setMap(1)
2. MapModel.setPos(0,1)
3. MapModel.moveRight()
4. RectModel.setRectColor(MapModel.getRoom());
5. Test if the color is blue in the first room
6. MapModel.moveUp()
7. RectModel.setRectColor(MapModel.getRoom());
8. Test if rectColor, the variable, and getRectColor() is equal
9. Test if the color is orange in player position (1,0)

Expected result:

- The color of the room should match the position that you have on the map.

Related user story:

- As a user, I want the room to change its color to correspond to the door I just clicked.
- As a user I want the color of the room to be the same on the middle screen as my position on the map.

Motivation:
This was a more extensive test to investigate if the color of the room corresponded with the position on the map every time the player moved.

RectModelTest.testGetRoomColor() - Test that the room color is set correctly

Test ID: T08
Test case description:
This test if the color of the room is set to the correct color.

Precondition:
Knowledge about that the colors for the room is stored in a HashMap where the colors are stored as a string and not as an int in getRectColor(). You also need to

understand that previously run tests have already set the color of the room to orange.

Test steps:

1. Compare the roomcolor of the set room with the corresponding string in drawMap.

Expected results:
This test should result in that the color is orange and it therefore return true.

Related user story:

- As a user I want the color of the room to be the same on the middle screen as my position on the map.

DoorModelTest.testSetDoor() - The doors should have the right color

Test ID: T09
Test case description:
This test investigates if we set a certain position on the map, the colors of the doors and their positions should be read correctly from the roomcode.

Precondition:
It is necessary to understand that each room is represented by a five numbered string, the first number represent the room, the second the door to the north of the room, the third the east door, the fourth the south door, and lastly the fifth number contain the information about the west door.

Test steps:

1. MapModel.setMap(1)
2. MapModel.setPos(1,0)
3. Call the method setDoor("13027”) with the doorCode of the room
4. Compare the number on every position with the right color

Expected results:
When calling the doors the correct door will be chosen and the correct door should have been asserted to it

Related user story:

- As a user, I want to see the room I am in, with the correct doors attached to it, as seen on the map.

Motivation:
We wanted to check if the method getDoor() returns the right colors of the adjacent rooms.

KeyModelTest.testKeys() – The Keys' functionality

Test ID: T10

Test case description:

This test should test the functionality of the keys.

Precondition:

Knowledge about the keys' class and how it is constructed.

Test steps:

- Not implemented yet

Expected results:

The keyes work as they are supposed to and therefore we will get true

Related user story:

- As a user, I want to be able to pick up the keys using the touch function.
- As a user I should only have access to a door if I have the correct key.
- As a User, I want to see the keys that I currently have in my inventory.

Motivation:

It is very important that the logic behind the keys work as they are supposed to in order to play the game.