Assignment 05

Group 02

Exercise 1 - Anonymous peer suggestions

The 15 improvements we decided to implement are:

1. Why is there only one level? -> implement multi-level system
2. Cannot click the two players menu button, so why do you have that button?
3. Pressing 2 in the help screen starts the singleplayer game and not the multiplayer game.
4. Make the images 2,3,4,5 more descriptive (maybe 2th-quadrant or something).
5. What is qix mp3, can be renamed to background-music.mp3, because qix.mp3 is a bit too vague.
6. Qixfont should be a resource, but is in the src/main/java folder in the parent package.
7. Be consistent and do for all classes and always do the following, not just in GameScene: @BeforeClass public static void BeforeClass(){ new JFXPanel();}
8. Remove commented out tests in SparxTest. https://github.com/Tim-W/SEMProject/commit/df0c008b224828c8ce10ba211b448fcfd8f9c138#diff-4a8beb3dabf6cff0afa1517d8bc8e7d0
9. Be consistent with your JavaDoc, see qixStixTest and CursorSparxTest, Javadoc does not start with Capital letter.
10. Remove commented code from the tests in cursorSparxTest in CollisionHandlerTest.
11. Remove commented out tests in SparxTest.
12. Remove the empty tests from unitTest at the bottom of the file.
13. Improve and extend javadoc of Game Elements as they are quite vague for someone who has never played or is not familiar to the game.
14. AreaTracker: Some private methods have comments in the code, while this can be javadoc. Change this to valid javadoc (`resetAreaTracker`, `setOuterBorders`, `updateScoreCounter`, `hitQix`).
15. Help page: add back button.

To view more information and view the progress of these implementations, refer to issue #266. https://github.com/Tim-W/SEMProject/issues/266

Exercise 2 - Software Metrics

The snapshot was taken of version V0.9. The result can be found in “docs/Sprint #5/SEM Group 2 Incode.result”.

a) We decided to fix the Cursor and Unit classes because their cumulative Severity was 4 (the highest). The flaws that were accounted are.

- The Unit class has a data clump, too many parameters in the constructor method. Also, the class **is exposing a significant amount of data in its public interface**, either via public attributes or public accessor methods.

- Cursor is a god class: The class **uses many attributes from external classes**, directly or via accessor methods. The class **is excessively large and complex,** due to its methods having a high cyclomatic complexity and nesting level. This class **is very non-cohesive,** in terms of how class attributes are used by its methods. The design choices leading to this are that Cursor just has a lot of properties, and a lot of things it needs to do (powerups, fuse, moving)
- The third was a cluster of flaws. All of our unit classes and subclasses were said to have a data clumb,

b) Fix the design flaw or extensively and precisely explain why this detected flaw is not an error and, thus, should not be fixed (10 pts).

- For the Cursor class, we managed to get the cumulative severity down from 4 to 3. We did this by making a fuseHandler and a powerupHandler, which made sure cursor did not to do all these things. Also, we reduced the complexity of a few methods. We did not manage to get the severity down to 0 however. Cursor is nested in so deep in our code, and it does many, many things. We don't think this is necessarily a bad thing though. Yes, cursor has many responsibilities, but those are all responsiblities you want to move. For example, the keypresses are handled in cursor, as only cursor knows where it can and can not move (as of now).

Exercise 3 - Walking in your TA's shoes

1. Requirements for the game extension:

Multi-level Requirements
Must-haves
- A player is able to play multiple levels.
- When a player reaches the target percentage a new level should be created.
- Units should respawn in the new level.
- Percentage should reset in the new level.

Should-haves
- The next level should have a higher difficulty.

Could-haves
- Additional sparx or qix could spawn.

2. UML class diagram of the multi-level implementation (**nl.tudelft.sem.group02.level** package):