CMPT 276 Assignment 3 Code Review

By Jay Bator and Rahul Naterwala

For this we were looking over the UI class that was created for rendering the game.

Code Smell 1:

Commit Before: e4cb1c586c89eaa2e39169224ebbcda69272a234

Commit After: 257a643c1500a422fefdb82157bad3f1202de808

The first thing we noticed is that on 86 a game object is instantiated. This seems to be some artifact left over from manual sanity tests made when first starting the UI class. We decided to remove this artifact.

Code Smell 2:

Commit Before: 257a643c1500a422fefdb82157bad3f1202de808

Commit After: f26deab958c472e9f8c8a5d5bd388acde6230efa

The next code smell we noticed is that in the RenderGame method for creating the score text we use numbers to multiply the cell width by some value to dynamically place where the score text is on the screen, and depending on what the score is the number we put into this function to render may be different. The same multipliers are used regardless of which branch is used, so it would be better to put them as constants that could be changed in one place instead of changing it multiple times. This branch we speak of exists on lines 141 to 147 in UI.java.

Code Smell 3:

Commit Before: f26deab958c472e9f8c8a5d5bd388acde6230efa

Commit After: 6f8623de6e64fce63b5c9535500ef8f89ad77cdd

The third code smell that we found is that in the constructor for the UI class multipliers are used to make sure that we properly render the size of the cells that are used to make up the maze, so for instance if the grid was to be 32 by 18 we would divide screen width and height by those numbers to get the values for this. These values are repeated numerous times while importing all of the objects into the application, so we made constant variables for these, so they could be changed in one place instead of many.

Code Smell 4:

Commit Before: 6f8623de6e64fce63b5c9535500ef8f89ad77cdd

Commit After: 2dfc7261fe9f35889cecedc7b2f9c4c83869c3d6

In the RenderMenu() method we use multipliers again to calculate where to render the playGame and howToPlay buttons. We will again move these multipliers to constant, so they can be changed in one place, but only for X, since the y value is different between the two but the X value will always be the same since they are both centered.

Code Smell 5:

COMMIT BEFORE: 2DFC7261FE9F35889CECEDC7B2F9C4C83869C3D6

COMMIT AFTER: 86D65B556860D3A4F69E2BD9DBFE4A49DF9ACEC5

In the RenderEntity() method within each for loop we repeat the same 2 lines of code. We made a RenderEntitySet() to do this instead of repeating code.

Code Smell 6:

COMMIT BEFORE: 86D65B556860D3A4F69E2BD9DBFE4A49DF9ACEC5

COMMIT AFTER: 32ACC3E2CEC63810115C56B48D43006FC90EDAE5

Our final code smell is that we realized for bringing in the images in the Image constructor the (false, false) at the end needed to be written every time for the image constructor, which could get tedious in the future for future update where we might need more Images, so we created ImportImage() method to make the syntax easier. Also, it allows for easier work for newer programmers since they won't need to figure out why we do the (false, false).

NOTE we were unable to test these refactorings at the time due to the sdks required not being installed on Jay's laptop that we were using. We found some interfacing errors later and fixed them in the following commit.

67166a47fbcd344b02df62665b93cc9623d0e79c