Bubble Beam - Assignment 4

Bavdaz, Luka 4228561
Clark, Liam 4303423
Gmelig Meyling, Jan-Willem 4305167
Hoek, Leon 4021606
Smulders, Sam 4225007

October 19, 2014

1 20-Time, reloaded

1.1 Implementation

Our requirements for the 20-time assignment is implemented, and can be found on our repository on the master branch.

1.2 Analysis and design phases

The analysis and design phase document is located on our devhub repository on the master branch, under "report/Assignment4 - Analysis and design phase document.pdf"

2 Software Metrics

2.1 Analysis file

Our inCode analysis file can be found on our repository on the master branch, under "inCode/shoot—1413576881531.result

2.2 Three design flaws

2.2.1 MovingBubble Dataclass

Design choices or errors leading to the detected design flaw
The MovingBubble is recognised as a Dataclass by inCode. This was mostly because one attribute, "velocity" was changed outside of the class by using a getter and a setter method. Too much of this small class was exposed in its public interface because of its public accessor methods.

Fix the design flaw
To solve this problem, we removed the unnecessary accessor methods. We will explain why the methods where unnecessary, and how we changed the class to be able to remove each method.

`setTruePosition(Vector2f)`

Firstly the method was unused. Secondly there is no reason for an other class to set the position of the bubble while it is already moving. The truePosition should only be set by an other class in the constructor, where the start location is set. For changing the position from within the class, the attribute itself can be directly accessed.

`getTruePosition():Vector2f`

Firstly the method was only used by tests, even though the tests should have been able to access the protected truePosition directly. The test should be in the same packages as the class to be tested to facilitate this direct accessing. Secondly, this method returns a pointer, and not a copy. This means that changing the received object of this method affects the truePosition of the moving bubble.

`getVelocity():Vector2f` and `setVelocity(Vector2f)`

The methods getVelocity and setVelocity made the velocity variable too accessible to other classes outside of the movingBubble. Both methods where used in the gameTick method of GameController to get the velocity of the moving bubble, adding some velocity to it and setting the velocity of the moving

bubble to the new calculated velocity. Because of this we replaced these methods with the method `addVelocity(Vector2f)`.

`getScreenSize():Dimension`

This method was only used in tests, but since the test itself gave the screenSize to the MovingBubble class in the constructor, the original variable can be used.

2.2.2 SlaveGameController

Design choices or errors leading to the detected design flaw

We made a distinction between `MasterGameController` and `SlaveGameController` for the multiplayer game mode. These game controllers extended the `AbstractGameController`, in which shared logic - and basically the game level logic - was stored. The distincition between master and slave was made to bind the correct event handlers to the controllers: the `MasterGameController` should be controlled by user input, and thus be bound to *MouseListeners*. The `SlaveGameController` is controlled over the network, by the other players `MasterGameController` which transmits its events over the network. That said, the `MasterGameController` and `SlaveGameController` actually did not add any logic to the `AbstractGameController`, it only added some event listeners.

Fix the design flaw

To solve this problem, we decided to completely remove the `SlaveGameController` and `MasterGameController` classes. The methods that are used to bind network adapters and event handlers are moved to the `GameController` - which is not abstract anymore. These are now methods which gets called from the place where the `GameController` is instantiated.

2.2.3 Another design flaw

Design choices or errors leading to the detected design flaw

Luckily, *InCode* only found above two issues (2.2.1 and 2.2.2). Therefore, we had to look in the code for further design flaws. One thing we've found, was that classes were quite dependent on each others static constants. This was an issue, because when for example you resize your screen, and want painted elements resized as well, these constants are not correct. It also made it difficult to change this values, if you wanted to change the game to have fewer but bigger bubbles, constant values in various classes had to be adjusted.

Fix the design flaw

We have fixed this issue by making all static constants either private or protected. Obviously, quite a few errors appeared. Then we changed the code accessing the old constants to use the attribute accessors instead.