Assignment 4

Group 66, Bust-a-move

Exercise 1 - Code improvements (30 pts)

1. Where do you use if or case statements in your source code? Refactor 10 of these statements used in different scenarios, so that they are not necessary anymore and describe your refactoring.
Instead of if or case statements you can use, for example, design patterns, polymorphism, or a technique called double dispatch.
If you have less than 10 of these if or case statements, you have less work to do!
If you find some rare cases of if or case statements that cannot be refactored you can convincingly explain why (we recommend to consult with your TA before going this way); these cases count toward the 10 you have to refactor.

The changes in the code that are made where done with respect to commit:
671a7a86cf9968477c28bde52658ca55f14484fb
https://github.com/iiKoe/bustamove/pull/108/commits/671a7a86cf9968477c28bde52658ca55f14484fb

In total the number of refactored if/else/case blocks is 11. These fall under three categories.

The first category is the change of the BallType enum. This enum can now be used to create a ball, so a new ball can be made depending on the type instead of case statements. Originally there was a if/else statement in the BallManager which would determine if a BombBall should be created depending on the type. And a if was used to get a ball from the The removed statements are in:

Removal #	**Class**	**Line**
1	BallManager	140
2	BallManager	160
3	BallManager	213
4	ColoredBall	46

The second category was the mute check and related components in the AudioManager. There a toggle of the audio was implemented using a if statement and a muted boolean. This is now changed to a state pattern. This also allows for new additional sound effects to easily be implemented.

Removal #	Class	Line
5	AudioManager	50
6	AudioManager	69
7	AudioManager	85
8	AudioManager	94
9	AudioManager	103

The third category was the removal of the case statements for both the popping and static ball animation instance. There were loaded from the assetloader depending on the ball type. These are now moved to the BallType enum. The Ball class holds an instance of this BallType so the correct animation can easily be retrieved.

Removal #	Class	Line
10	Ball	267
11	Ball	337

Exercise 2 - Teaming up (30 pts)

1. In this exercise, you decide together with your TA, during the group meeting on Monday, new game features to add to your game.
After you decide on the tasks, write a requirements document, which will be evaluated in the same way as for the requirements document of the initial version. Afterwards you must implement the requirements.

Teaming up related requirements can be found within the assignment 4 Teaming up directory.

2. During the analysis and design phases of this extension use responsibility driven design and UML (push to the repository the single PDF file including all the documents produced).

Teaming up related requirements can be found within the assignment 4 Teaming up directory.

Exercise 3 - Walking in your TA's shoes (30 pts)

Reading, understanding, and evaluating code written by other software engineers is the bread and butter of professional software engineering. Ask your TA for the codebase of another group (which will be provided as an anonymous .zip archive) and perform the following tasks:

1. Grade the code quality according to the rubrics. You have to deeply detail the reasons for your choices.
All details can be found in a separate attached document entitled "a4ex366__code_quality" in the walking in your ta's shoes directory.

2. Propose meaningful enhancements to the other's group codebase. Meaningful enhancements are, for example, based on design patterns and design principles. These enhancements must be worth 30 points for a weekly assignment.

Hello unknown team. Your assignment for this week is to implement a few weapon upgrades. Because of your PowerUpBehaviour interface and ShootBehaviour interface, it is easy to create a new weapon powerup.
A couple of our suggestions are triple beam shots, a burst, continuous fire, side-shooters, shooting in 4 directions (forward, backward, left, right), a targeting missile and maybe a one-time shot that shoots in all directions.
Discuss with your TA which of these you would like to do.
If you need more inspiration, there is an awesome game called Grid wars.

A few unrelated things that we would like to point out. If you decrease the deceleration by half, it would give the game a more space-y feeling. Also, the current space ship images do not give a good indication when you have shields (we had to look up that it was red).