

## Assignment 4

### Part 1

There are three class and method level metrics that we improved upon. They are coupling, length of the methods in the class and lack of cohesion.

The reason we decided to improve coupling was that we wanted to reduce coupling so that the reliance on the methods of a class to the methods of another class is reduced. This makes it so that further improvements are easier to make. In this way we wanted to improve the inter-module complexity. Because if we changed a method during the time that the coupling was not reduced, we needed to change a lot of other methods that relied on this method. Therefore we reduced the coupling of methods so that we would not have to put in so much work when we made a little change in our method.

The reason we decided to improve on the size of the method, was because methods which were too long were difficult to get a good overview of them, and very difficult for developers who didn't create them in the first place to understand how they work or how to modify them. That is why we improved the methods by splitting the methods into two separate methods, which makes the method much easier to understand and does not give the method too much tasks on his own. Also it builds upon an important pillar of software engineering which is separation of concerns which ensures that each method has one and only one goal. Which in the end contributes to making testing and modifying methods easier.

We wanted to improve the problem with the lack of cohesion is to improve the intra-module complexity. This will ensure that the method in a class belongs together and they are more understandable. That in turn will help so that whenever a new developer looks at the code and wants to modify something all code related to whatever he's working on will be easy to find.

### Part 2









#### Class

List of all classes (#22)









ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
----	-------	----------	------------	------------------	------	-----	------------	----------	------------------	------

1	Asteroid					123	low-medium	medium-high	medium-high	low-medium
7	Asteroid					123	low-medium	low-medium	medium-high	low-medium









For the asteroid class, we improved the coupling and the size of methods because we reduced the interconnection to other modules. We put everything we possibly could related to asteroids in the asteroids class.

List of all classes (#22)										
ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
3	Player					75	low-medium	medium-high	medium-high	low-medium
14	Player					37	low-medium	low	low	low

For player we wanted to reduce the coupling and the lack of cohesion. That's why we build an abstract class called entity to reduce the lack of cohesion as all the entities are now related to each other. For the coupling, the player only calls the class Bullet when shooting, but when a change is made in the bullet it does not have an effect anymore on the player class as every bullet, as you just add objects to the arraylist.



List of all classes (#22)										
ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
3	MainMenuScreen					51	low-medium	medium-high	low	low-medium
4	MainMenuScreen					49	low-medium	medium-high	low	low

For MainMenuScreen we improved the size by removing unused imports and unused attributes inside the method and inside the class. Also, we reduced unnecessary checks in the methods.

List of all classes (#22)										
ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
2	SignUpScreen					72	low-medium	medium-high	low-medium	low-medium
5	SignUpScreen					34	medium-high	low-medium	low	low

We wanted to reduce the lack of cohesion for SignUpScreen, by making another abstract class called authentication, because the render methods of both loginScreen and signupScreen use the same render method. So by building an abstract class called authentication and let both of the classes extend from there the cohesion will go up, as the two classes relate to each other through the abstract class.

Also the size of the class is smaller, because you put the method in the abstract class. By doing this duplicate code is removed as the method is just called from the authentication class.

List of all classes (#22)										
ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
3	LoginScreen					69	low-medium	medium-high	low-medium	low-medium
6	LoginScreen					30	medium-high	low-medium	low	low

We wanted to reduce the lack of cohesion for LoginScreen, by making another abstract class called authentication, because the render methods of both loginScreen and signupScreen use the same render method. So by building an abstract class called authentication and let both of the classes extend from there the cohesion will go up, as the two classes relate to each other through the abstract class.

Also the size of the class is smaller, because you put the method in the abstract class. By doing this duplicate code is removed as the method will now be called from the authentication class.

## Methods

We have decided to focus mainly on the issues of lack of cohesion in the asteroids class. We fixed this class by changing the (1) constructor and the (2) Update method. Firstly, we split the constructor into three parts, firstly, the constructor itself, then we called the createVertices method and the generateVelocity method inside the constructor. That also makes it easier since it splits concerns by assigning each task to a method to do it. Secondly, we did the same for the update method. Inside the update method, we called the checkWrap and createVertices method. What these did were reduce coupling and lack of coherence.

The alivescreen class is a major part of the application since that's where the user will mostly be. So we needed it to be well developed. To that end, we tried to make the code as clean as possible by creating a method for each task and try to make it as small as possible. This of course works towards our goal of reducing method sizes and lack of coherence. An added benefit is that if we wanted to support multiplayer we would only have to change a few parameters using the newly created initPlayer method.

(3) An example of what was changed is a method we had called asteroidCollision which we refactored to be three separate methods with separate concerns called checkAllCollisions, checkPlayerEnemyCollision and checkPlayerEnemyCollision.

(4) In the AuthenticationScreen the show method was huge and included a lot of functions (configuring buttons displaying content and initializing resources) which was very hard to maintain. Another problem was that configuring the buttons is needed for more than one class (LoginScreen, SignupScreen) so there was code duplication. This is problematic because at some point if someone would make a change to how the buttons are configured in one class and not the other it would cause a lot of bugs. And in the long run if more classes use the same configuration it would only make the problem worse. For that we decided to split the show method into show and initButtons, where init buttons was contained in the superclass of LoginScreen and SignUpScreen, namely AuthenticationScreen. This solves both problems, changes only need to happen in one place hence increasing cohesion, bettering code size and functionality and reducing code duplication.

(5) Since the LoginScreen class inherits from the Authentication screen class, we implemented the an abstract method (initButtons) of the Authentification screen class in the loginscreen class. We implemented the implementation of the abstract innitButtons in the loginscreen. In this way we improved cohesion. This also reduced code duplication, improved method size and improved cohesion.