Space Invaders : Sprint 2 Assignment

Group 22
TI2206 Software Engineering Methods

Ege de Bruin
Bryan van Wijk
Dorian de Koning
Jochem Lugtenburg

September 25, 2015

Supervisor: Dr. A. Bacchelli
TA: Danny Plenge

Delft University of Technology
Faculty of EEMCS

Excercise 1 - 20-Time

To improve our code, we want to split several classes into more classes. The classes we want to split are Game and GameUIController, because these classes are now too big in our code. We have chosen to split these classes in the following classes:
Game:

Controller

- UnitController
- MovingUnitController
  - ExplosionController
  - BarricadeController
  - AlienController
  - SpaceShipController
  - BulletController
    - AlienBulletController
    - SpaceShipBulletController

Unit

We also want to change the Unit section of our code, we want to have an interface movable that is implemented by the movable/moving. This prevents a lot of duplicate code. Implementing this change would mean we would have to change all of the following classes:

- Unit
- Movable
  - Alien
  - SpaceShip
  - Bullet
    - SpaceShipBullet
    - AlienBullet
  - Explosion
  - Barricade

GameUiController

Changing the structure of unit in our code we also need to change the ui section. This will be the new class hierarchy. GameUIController:

- UiElement
  - UiElementUnit

        * UiElementAlien
        * UiElementSpaceShip
        * UiElementBarricade
        * UiElementExplosion
        * UiElementBullet
            - UiElementSpaceShipBullet
            - UiElementAlienBullet
    - Score
    - Lives

Since we did implement equals methods in most of (the suitable) classes but we did not implement hash code yet. Hashcode should be implemented too.

Exercise 1 UML

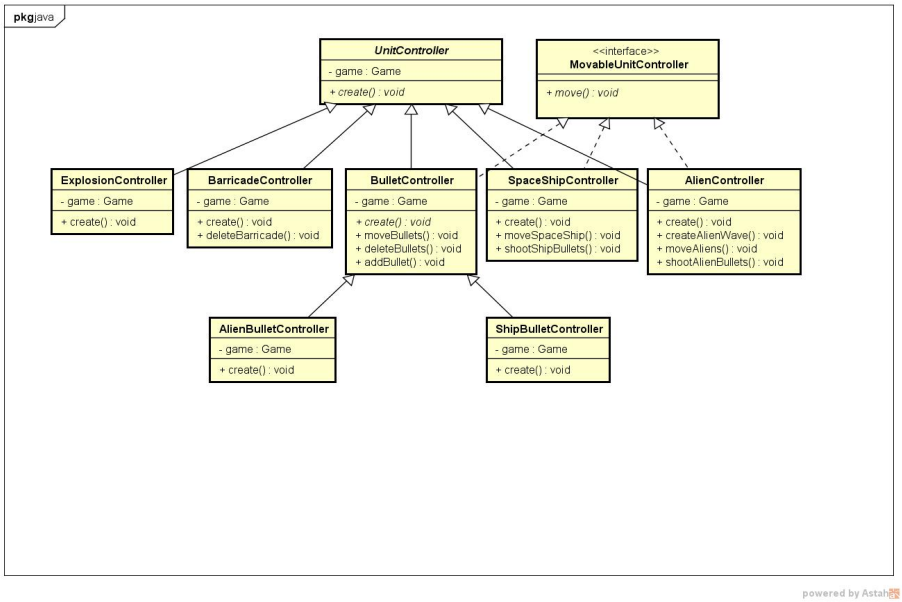

Figure 1: Controller UML Class Diagram

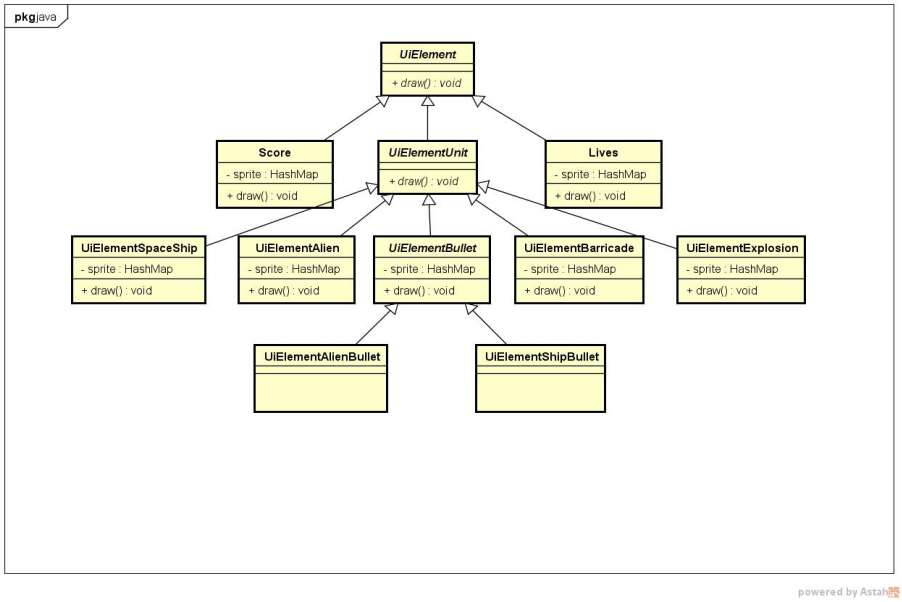

Figure 2: UIElement UML Class Diagram

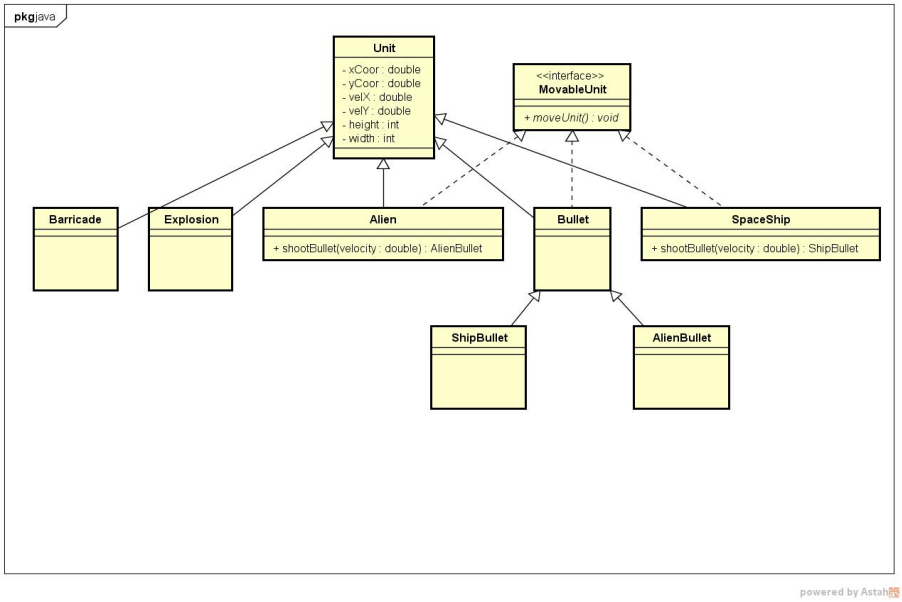

Figure 3: Unit UML Class Diagram

Code Improvement Functional Requirements

A list of functional requirements considered for code improvements using the MoSCoW method described in the previous section.

Must Haves

The Code Improvements must meet the following requirements:

- Code shall be Checkstyle compliant.
- Code shall be PMD and Findbugs compliant.
- Classes shall only have one responsibility.
- Methods shall not be too long.

Should Haves

The Code Improvements should meet the following requirements:

- Code shall contain Javadoc comments explaining complex methods.
- Code shall not be commented.
- Static shall be used where appropriate.
- Private access modifiers shall be used where appropriate.
- Test coverage shall be at least 75%.
- Code shall contain less duplicate code.

Could Haves

The Code Improvements could meet the following requirements:

- Variables shall be named in correct English.
- Methods shall be named in correct English.
- Classes shall be named in correct English.
- HashCode methods shall be implemented correctly where equals is used.

Would/Won't Haves

The Code Improvements won't meet the following requirements:

- ...