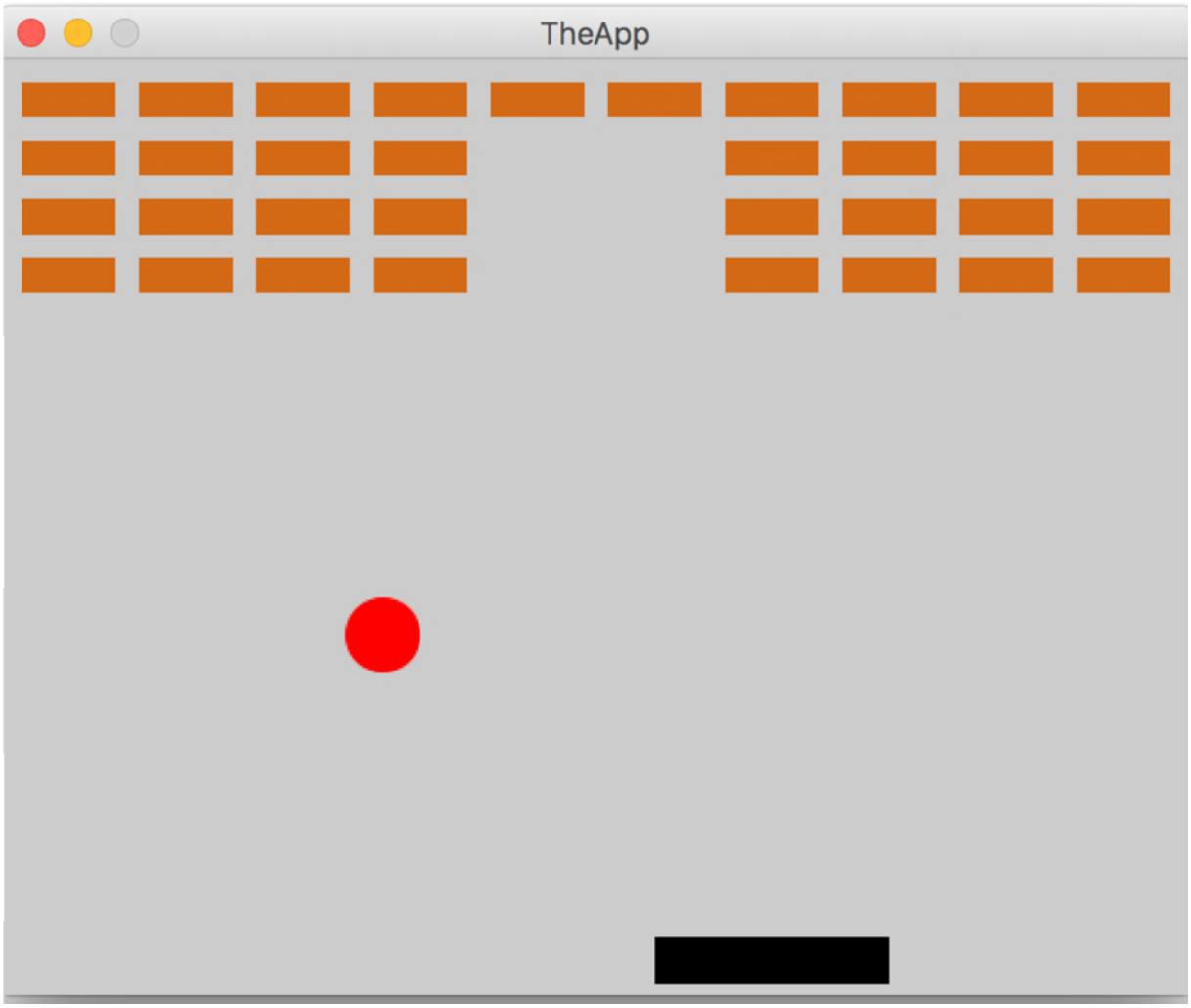
# OpenHPI - Java Capstone Series Pt. 1

# Documentation

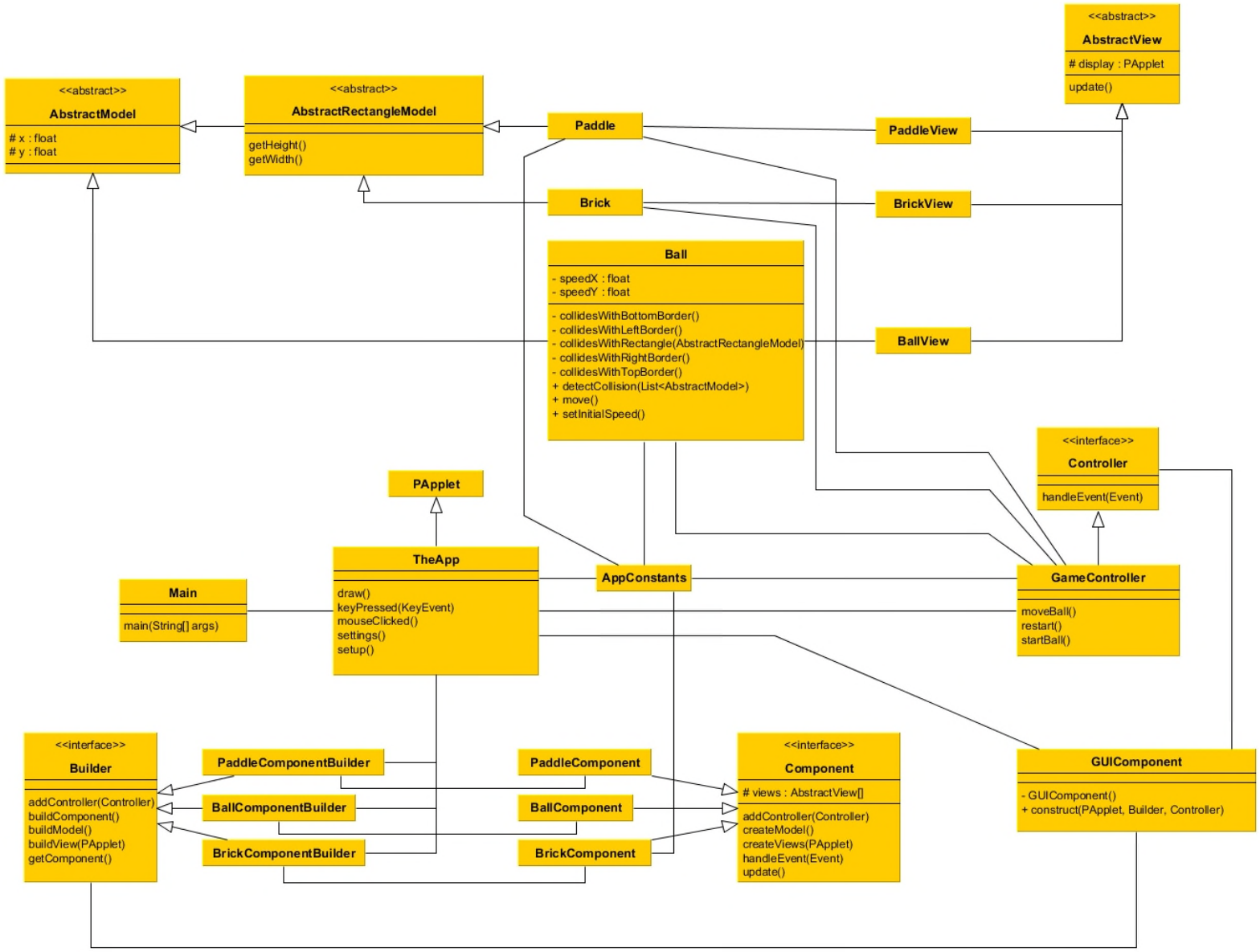
In the following I want to present my project. As you can already see in the screenshot below, I implemented BreakOut. At page 3, you can find the corresponding UML diagram for the code of the project.

My BreakOut implementation is more classical, I preferred to keep it simple like in a classical arcade game in the good old times ;) The game consists of the paddle and ball, obviously and a varying number of bricks, the number of bricks is calculated by the initial screen size.

The game works as expected. The paddle is moveable and reflects the ball, if they collide. The bricks disappear, if they are touched by the ball. The bricks also reflect the ball. The goal of the game is to remove all bricks. If the ball hits the bottom of the game area, the game is lost.



## UML class diagram for the project



## Introduction how to play the game

This section describes how to control the game.

| Key | Action |
|---|---|
| ESC | close the game |
| RETURN | start a new game |
| SPACE | pause the game |
| ← | move the paddle to the left |
| → | move the paddle to the right |

# Lab Report

This report describes  how the project evolved, which steps I toke, when, which problems arised and how I solved them.

## Setup

The first step consists of downloading the initial app source code from the GitHub project. The import into the Eclipse IDE was no problem. Eclipse indicated clearly the way to go by showing the compilation error in the Problem view. So I had to download Processing and extract the needed JAR file. The compilation error disappeared after the linkage of the missing lib in the Build Path of Eclipse.

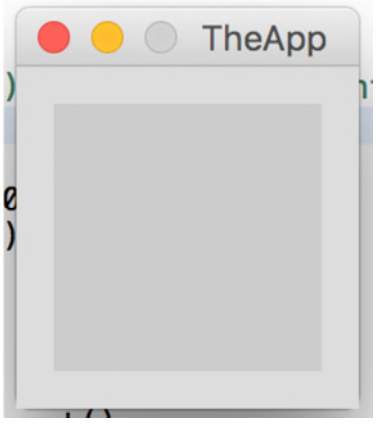
A bigger pitfall was the depency of Java 8. Before I used only Java 10, but the Processing lib is not compatible with Java 10.

```
java.lang.NoClassDefFoundError: com/apple/eawt/QuitHandler
    at java.base/java.lang.Class.getDeclaredMethods0(Native Method)
    at java.base/java.lang.Class.privateGetDeclaredMethods(Class.java:3139)
    at java.base/java.lang.Class.getMethodsRecursive(Class.java:3280)
    at java.base/java.lang.Class.getMethod0(Class.java:3266)
    at java.base/java.lang.Class.getMethod(Class.java:2063)
    at processing.core.PApplet.runSketch(PApplet.java:10716)
    at processing.core.PApplet.main(PApplet.java:10513)
    at Processing.main(Processing.java:6)
Caused by: java.lang.ClassNotFoundException: com.apple.eawt.QuitHandler
    at
java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java
:582)
    at
java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.j
ava:185)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:496)
    ... 8 more
```

So I had to install Java 8 again and select this JDK for the compilation of the project, which solved the second problem.

## First run of the application

After completion of the setup, the game was ready to use. Mh … maybe not, but at least it showed a grey canvas, which was the starting point for everything else.



## Creation of the paddle

After setting up the application and understanding the already given code, I started with the enlargement of the game canvas and the creation of the paddle to control the game.

I introduced the graphical representation of the paddle, which is a simple rectangle. I used for this the model-view-controller pattern. Hence, I created the model, which is represented through the class `Paddle` and the view, `PaddleView`. The elements of the game are all created with the builder pattern, which means that a component and a builder belong to every element. For example the paddle is constructed through the `PaddleComponent` and the construction process is controlled by the `PaddleComponentBuilder`.

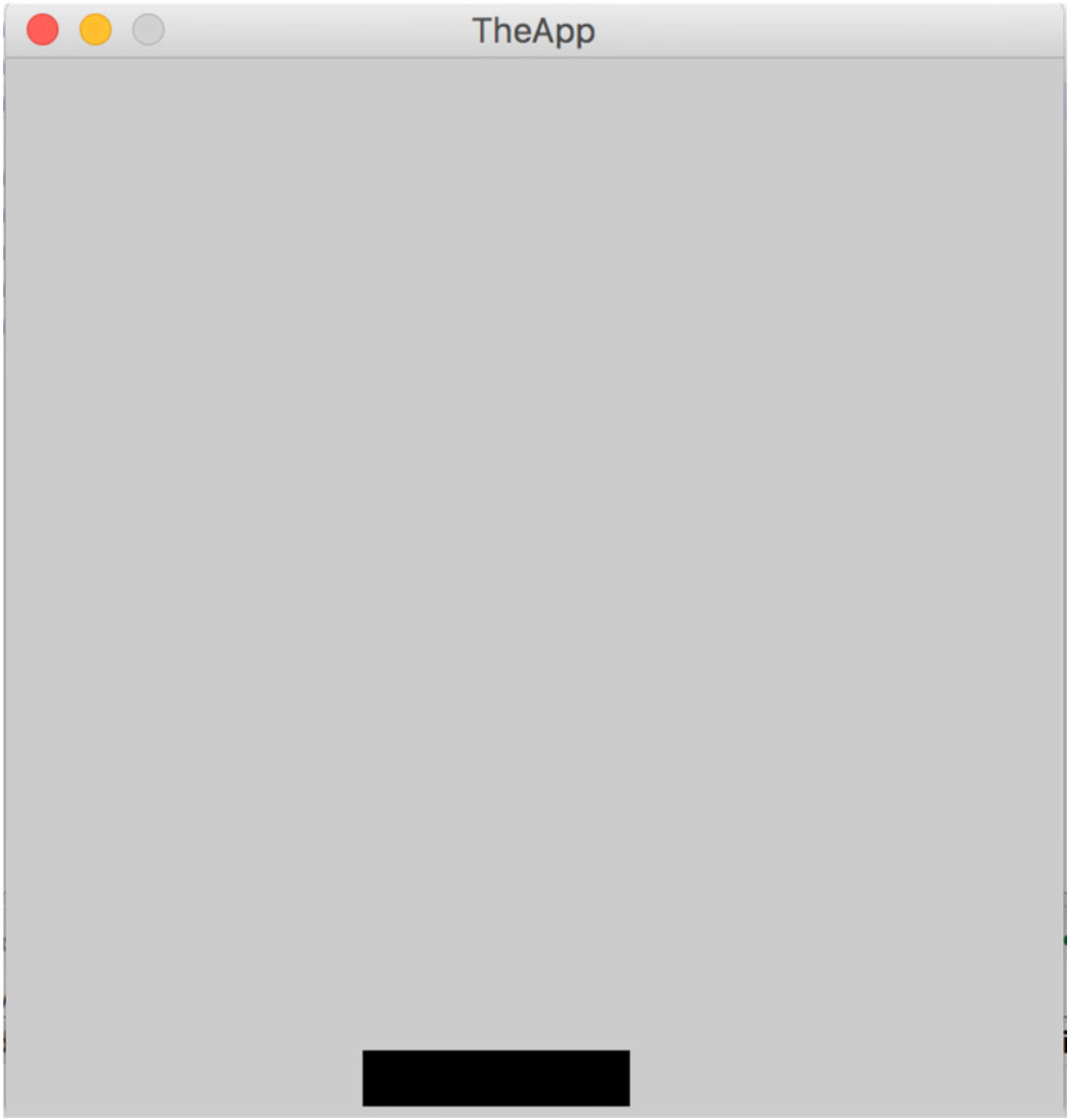



Figure 1 Visualisation of the paddle

## Movement of the paddle

The next step was to move the paddle. I used the `keyPressed` method of the `PApplet` class inmy `TheApp` class. This class is now listening for key events like in the Observer pattern and gives the information to the `GameController`, which reacts to the event with the `handleEvent` method. The mehod tells the `Paddle` to change their coordinates. The change is then reflected in the next frame, when the paddle is repainted by the `PaddleView`.

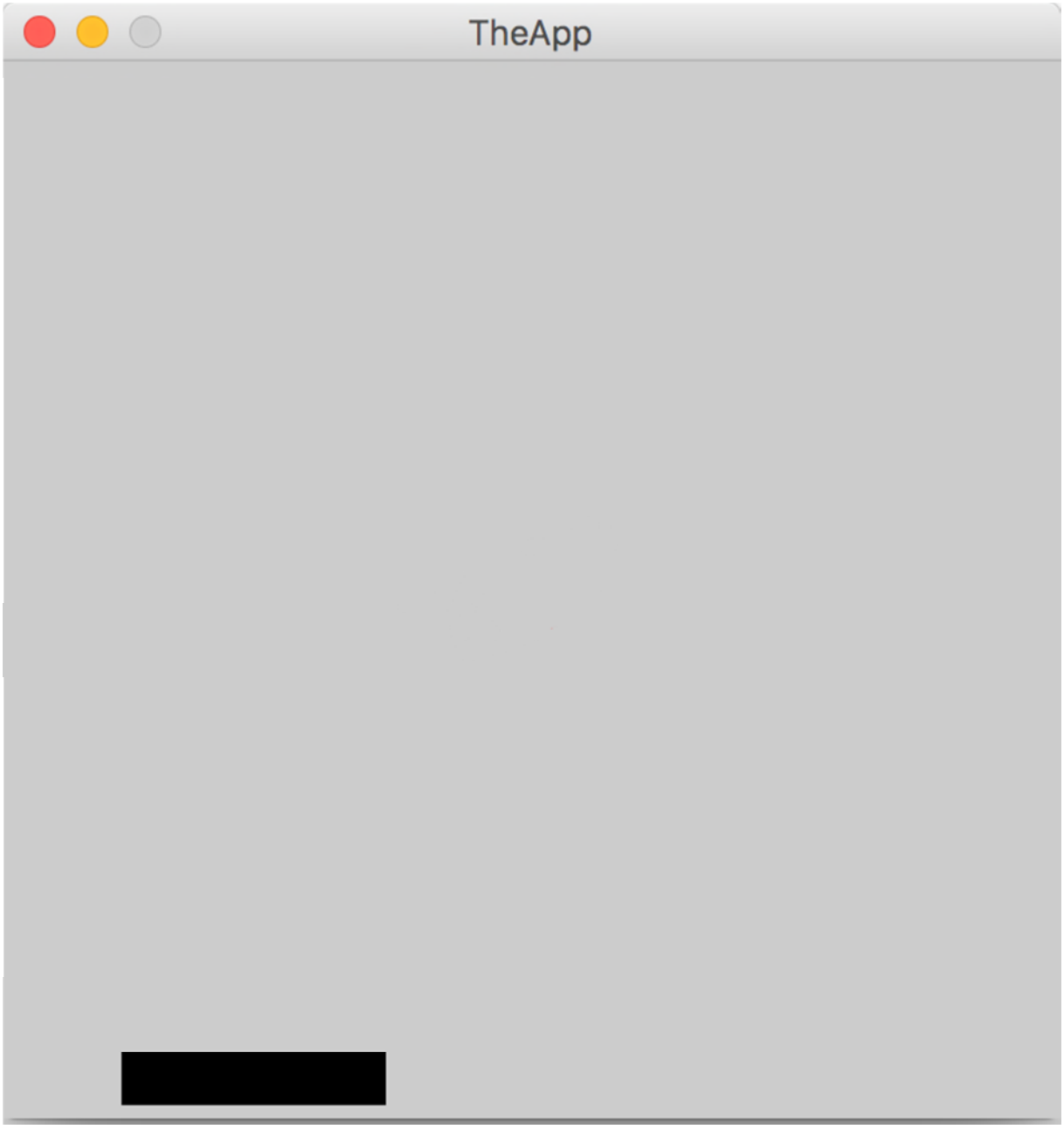



Figure 2 Movement of the paddle

## The ball

The central element in this game is the ball. I displayed the ball similar to the paddle with the classes `Ball` and `BallView`. The ball element is constructed by the `BallComponent` and the `BallComponentBuilder`.

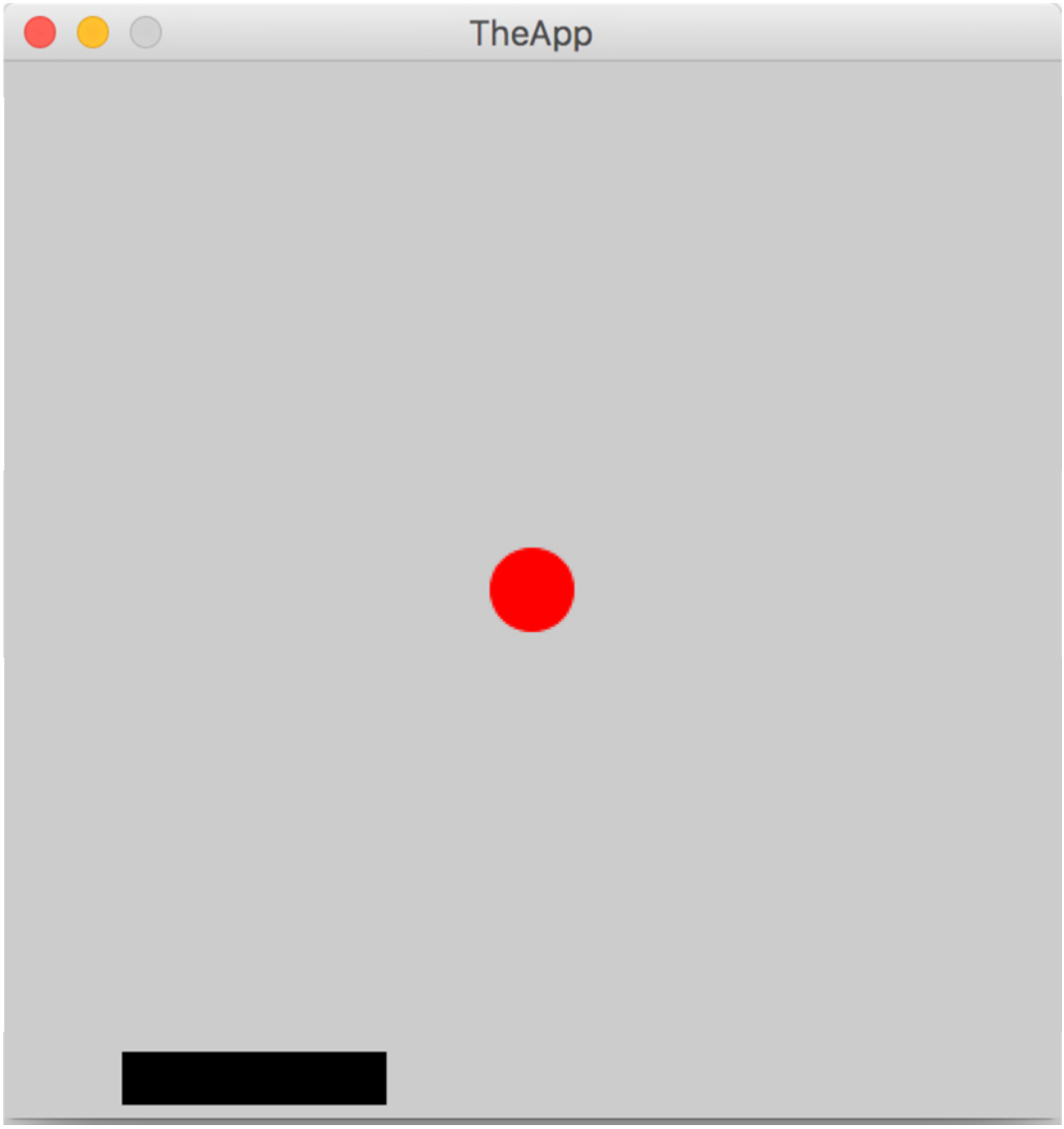



Figure 3 Visualisation of the ball

For me, the most challenging part of this game was the movement of the ball and the reaction to collisions with other game objects. The ball is self-responsible for the detection and the handling of collision. Every time when the frame changes, the ball calculates its new position and looks if it collides with the border of the game area or with the paddle. If the ball collides with a border, its direction is reversed. Despite the bottom border, then the ball stops and the game is lost. The collision handling for the paddle was a little bit more complicated, because if the direction would be reversed only, the ball would take the same paths endlessly and the user would have no option to steer the movement of the ball. So I changed the reflection angle of the ball depending to the intersection point between ball and paddle relative to the middle of the paddle. The further away the ball is from the center, the smaller the reflection angle becomes.

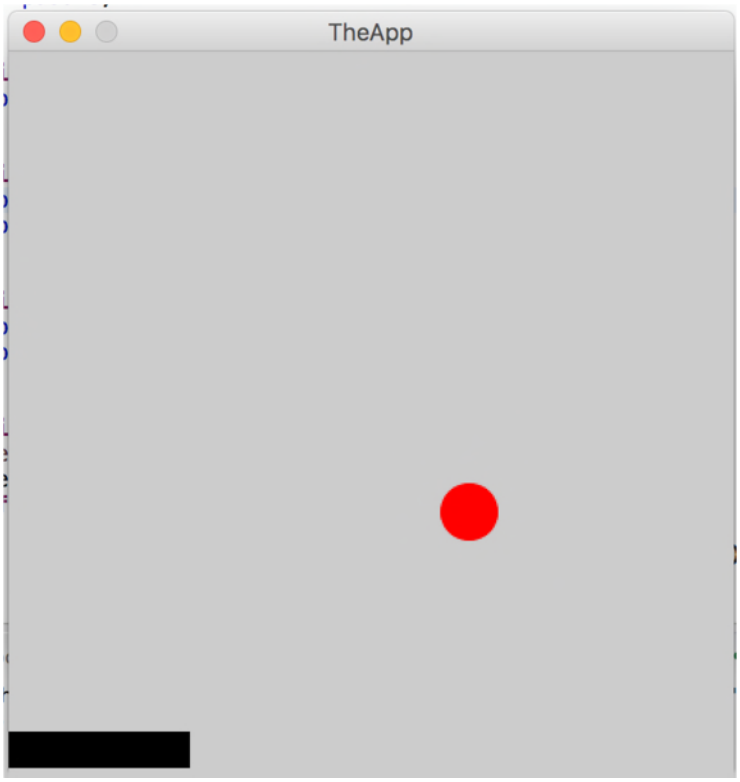



Figure 4 Movement of the ball

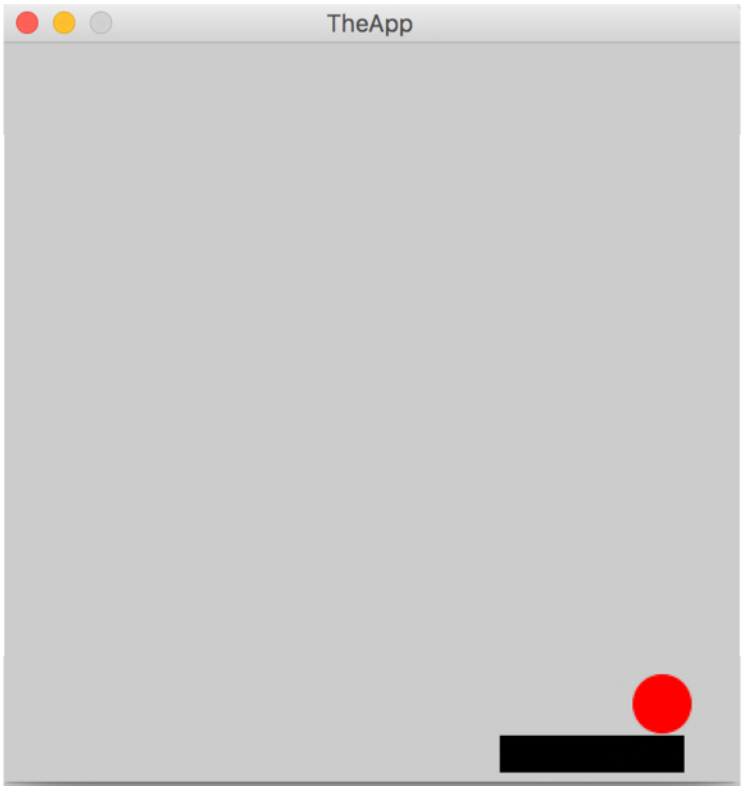



Figure 5 Collision detection between ball and paddle

## Adding of bricks

The last game elements were the bricks. The view element of the bricks is very close to the paddle, because it is rectangle with a different color. The bricks cannot move, they only vanish, if they are hit by the ball. I decided to make the brick only invisible. This has the benefit of just making them visible again, when the game is restarted. The bricks are constructed with the `BrickComponent` and the `BrickComponentBuilder`. They are displayed by the `BrickView` and the model isrepresented by the `Brick` class.

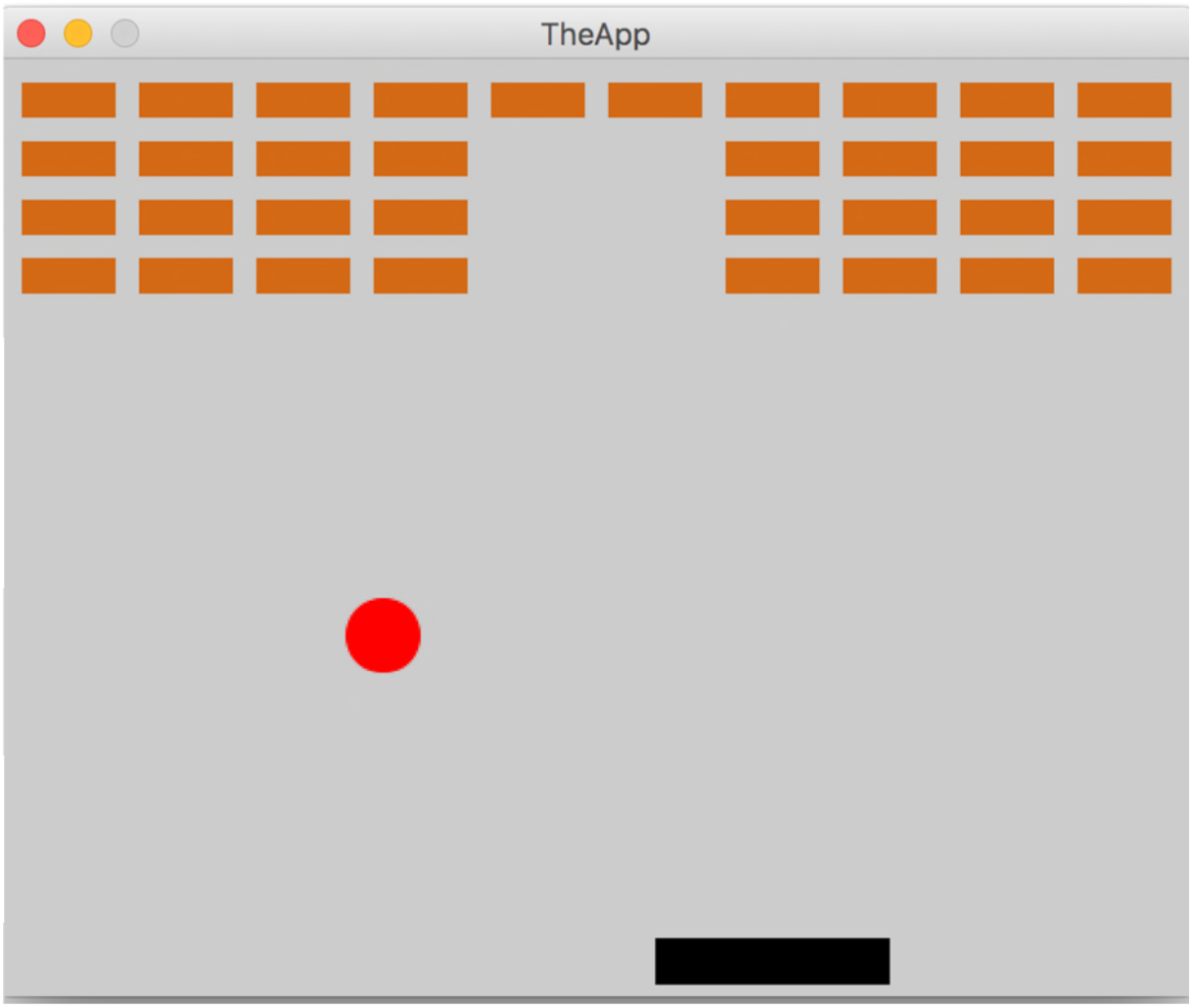



Figure 6 Introduction of the bricks

## Future work

In the future I would like to add a counter for points and maybe a way to vary different settings for the game.