Our version of MVC

In the game One More Cookie a modified version of MVC is used to structure the code. The library used, libgdx, makes it complicated to follow the classic MVC because of the already existing library classes that our classes extends and implements. The library is built in a way that our program has to adapt to for it to work in an effective way, not always following the MVC pattern.

Importing the library class Vector2 in our model classes

A traditional model class should be independent of the libraries and other external components of the program. In the our program some of the model classes are allowed to import the library class Vector2. A Vector2 is a 2D vector, a pair of floats, that are mainly used to describe positions and sizes in a 2D world. our model classes are suppose to store positions that are later used to create physical objects and graphic representations in the controller and view classes. In normal case the solution would be to write our own class Position that takes parameters (float x, float y). The problem is that the Position would still have to be converted to a Vector2 or we would have to separate position.x and position.y and send them in as individual floats in any future methods outside the model. Therefore we decided to just use the simple Vector2 in the model classes.

Render method in the Controller class (InGameController)

Our class InGameController implements the library class Screen. the InGameController is where the game itself, the running-from-the-enemy-and-eating-cookies-thing, is played out. The class Screen has a bunch of methods, like render, that needs to be implemented by our class implementing Screen. Render is simply an update method that are called every time the game updates, several times every second. Since a lot of the things that should happen every time render calls is clearly a method that belongs in a controller, it made more sense to implement Screen in the InGameController since the view should not call the controller. the “rendering” of the graphics are still done in the view classes and just called in the render() method in InGameController.

Screens

In our project there are six packages. The normal mvc-packages are the model, view and controller packages. Then we have the utils package containing some classes for converting meters to pixels, reading a tmx-file and a factory for creating physical bodies. The io package is similar to the utils. Then we have the package screens. The screen-classes are like a combination of view and controller. They are used for the different menus in the game and does not follow the mvc pattern since it does both the job of a controller and a view. But we decided to keep them this way since it works for us.