Microrécif

Ferrulli Massimiliano, Waldorff Carl Johan Træholt

April 28, 2024

1 General structure

When approaching the task specified in the project the code has been written following the principles of the Model-View-Controller (MVC) software design pattern. Structuring the code according to MVC the pattern is an effective way of organizing a code where a user interacts with large datasets. Moreover, it provides a practical framework for applying key principles of object oriented programming such as encapsulation, inheritance, polymorphism and abstraction. In the following we provide a detailed description of the code structure in the model.

2 Model

Model is the element tasked with reading and processing data on each individual *lifeform* in the microbial simulation as well as the status of the simulation itself. The subsystem is structured such that the module **Simulation** is the sole entry point of the data. In this module a class called Simulation is defined with class vectors (AlgaeVect, CoralVect, ScavengerVect) that contain unique_ptrs pointing to instances of the class *lifeform*. The 2 methods readFile and LineDecoding populates the object's attributes with instances of *lifeform*. Furthermore, methods to amend further instances of *lifeform* are declared with the prefix,"create-" and "set-", i.e. setAlgae[1] and createAlgae[2]. This way the attributes in simulation are modified during the course of the simulation. Additionally Simulation features methods to verify whether the data read satisfies certain criteria. This is handled by the methods with prefix, "check-", such as checkIntersect[3]. The further details of the methods are defined in other functions belonging to modules at deeper levels of the dependance network.

3 Data structure

The details of each *lifeform* is read and sorted into three subcategories as dictated by the text file provided. The sub classes are created using a class hierarchy in which the parental class is *lifeform*. In the *lifeform* class, the type of data that can be attributed to all individuals of the class is stored. In the case of our program it concerns the age and position of each individual. Moving a step down the hierarchy, the three subclasses, *Algae*, *Coral* and *Scavenger* inherit all attributes and methods of the *lifeform* class. In addition, they individually feature a unique specialization. The most simple is the *Algae* merely specialized by its name. Moreover, is

[1] l. 132, simulation.cc
[2] l. 144, simulation.cc
[3] l. 224, simulation.cc

Scavenger which additionally has an attribute for its radius, targetCoral and scavengerStatus. Lastly, we have *Coral* which distinguishes itself by containing a vector with position, angle and length of each of its segments along with attributes for id, coralStatus, developpement status (statusDev) and rotation direction (dirRotCor). In each class methods to extract data have been coded in order to verify that the attributes satisfy certain criteria. These methods are characterized by the prefix, "check-", such as checkRadius[4]. Other methods are able to provide data on the attributes within a class and they have the prefix, "get-", such as getLength[5].

4 Shape module

Utilizing the principle of encapsulation the type of the attributes for position and segments are defined utilizing the module **shape**. This module acts as an (almost) independent entity capable of describing geometric shapes such as rectangles, circles and segments and points. All types are defined in the interface as classes respectively, Rect, Circle and Segment, with the exception of S2d defined as a structure for position coordinates. Also in the interface there are the prototypes of various functions useful for determining the relationship between different Segments (intersection or superposition). Applying the principle of abstraction the definition of aforementioned functions are detailed in the implementation. Additional functions are capable of drawing the geometric shapes using the GTKmm library and further functions in the **graphic** module not further discussed in this report. Therefore, the **shape** module is not entirely independent.

[4] l. 106, lifeform.cc
[5] l. 45, shape.cc