Architecture 2D-GC visualization

Baksteen, Sarah
S3145034

Brink, Wouter
S3862348

Gros, Oane
S2972778

Jager, Maarten de
S2957906

Jong, Michiel de
S2550768

Strik, Oliver
S3100693

Client

Rohrbach, Léon
Figueiredo, Monique

TA

Argyrousis, George

Software Engineering
University of Groningen
March 25, 2019

Version 1.0

Introduction

The product we are developing is a tool to be used for the visualization and processing of 2D-GC data. This data is obtained through the process of gas chromatography and can give insight in the specific compound being analyzed. To effectively perform the analysis of a certain compound the ability to visualize the data in multiple dimensions is required. As 3D and 2D visualization of the data is deemed to have priority over the 1D visualization, the latter will not be part of the product for now. In order to view the visualized data in greater detail the product allows for zooming and moving within a graph. In addition the product enables the labelling of peaks and the integration over selected data.

Architectural overview

The general architecture is based on the MVC-model where the architecture is divided into 3 main components. In the most general case of an MVC-architecture there exists a controller with which the user interacts, the controller consequently manipulates the model component according to the input given by the user. The model then updates the view, which in its turn is shown to the viewer. For this particular program the MVC-model is slightly modified and a broad representation is given by figure 1.

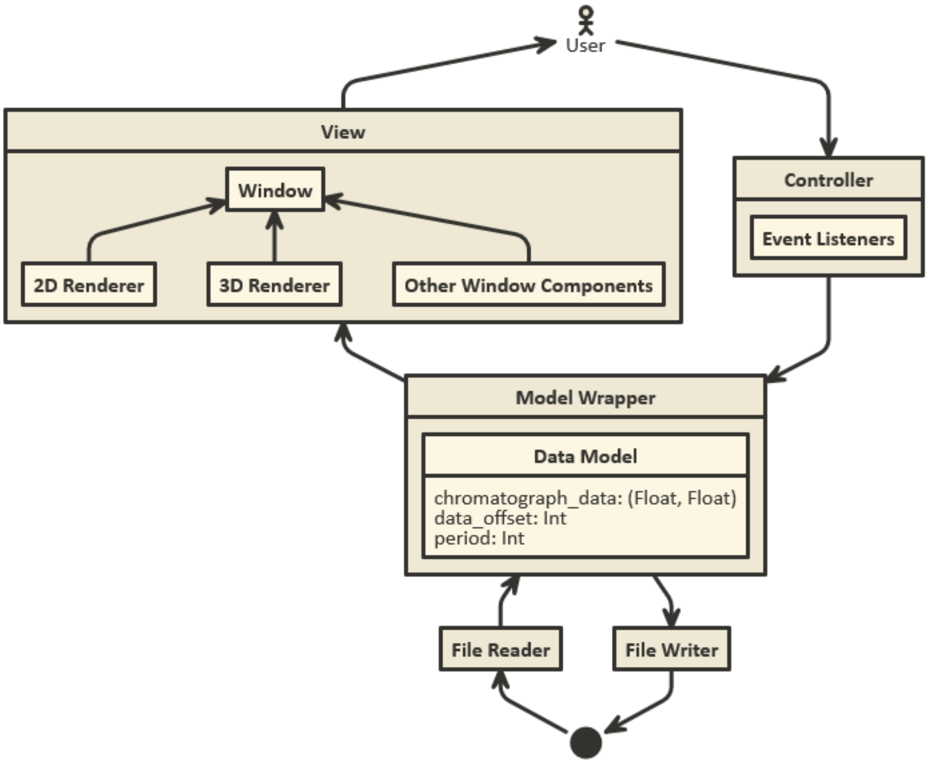

Figure 1: Diagram representation of the MVC-model used

The view component represents what is shown by the program on the main window. As requested the main window contains a 3d-visualization of the gc data and a 2d visualization of the 2d-gc data alongside the 3d graph. For now the architecture does not include the 1d representation and most of the functionality to be included in the 'other window components' is not present yet. Functionality such as zooming and moving within the graph is handled by the libraries which generate the visualization of the data themselves.

The model wrapper contains the model, that is in this case, the data. Since the user needs to be able to save peak data and corresponding information on integration values and labels and such it is necessary for the model to be able to write files.

Technology stack

The product is build on Python 3 software, which due to it's many available libraries and packages concerning data visualization and manipulation, makes for an convenient implementation. The used libraries include:

- **mayavi**
- **matplotlib**
- **mpl_toolkits**
- **tkinter**
- **numpy**

Date	author	description
19-03-2019	Maarten de Jager	initialized the document