# Project “Drawing editor for La Ruche”

Damir Buskulic, Université Savoie Mont Blanc

## La Ruche, introduction

La Ruche is an interactive editor of exercises for the [WIMS educational platform](http://wims.auto.u-psud.fr/wims/) (see also [the WIMSEDU association](https://wimsedu.info/?page_id=68). The WIMS platform offers a rich panoply of resources and exercises with free access or for personalized study. The pedagogical resources include

* exercises with automatic feedback and correction, which allow the learners to work at their own pace;
* interactive exercises with randomly assigned data, which allow the learners to work on the same type of exercise but with different data and as often as they wish;
* many other functionalities not relevant for our purpose…

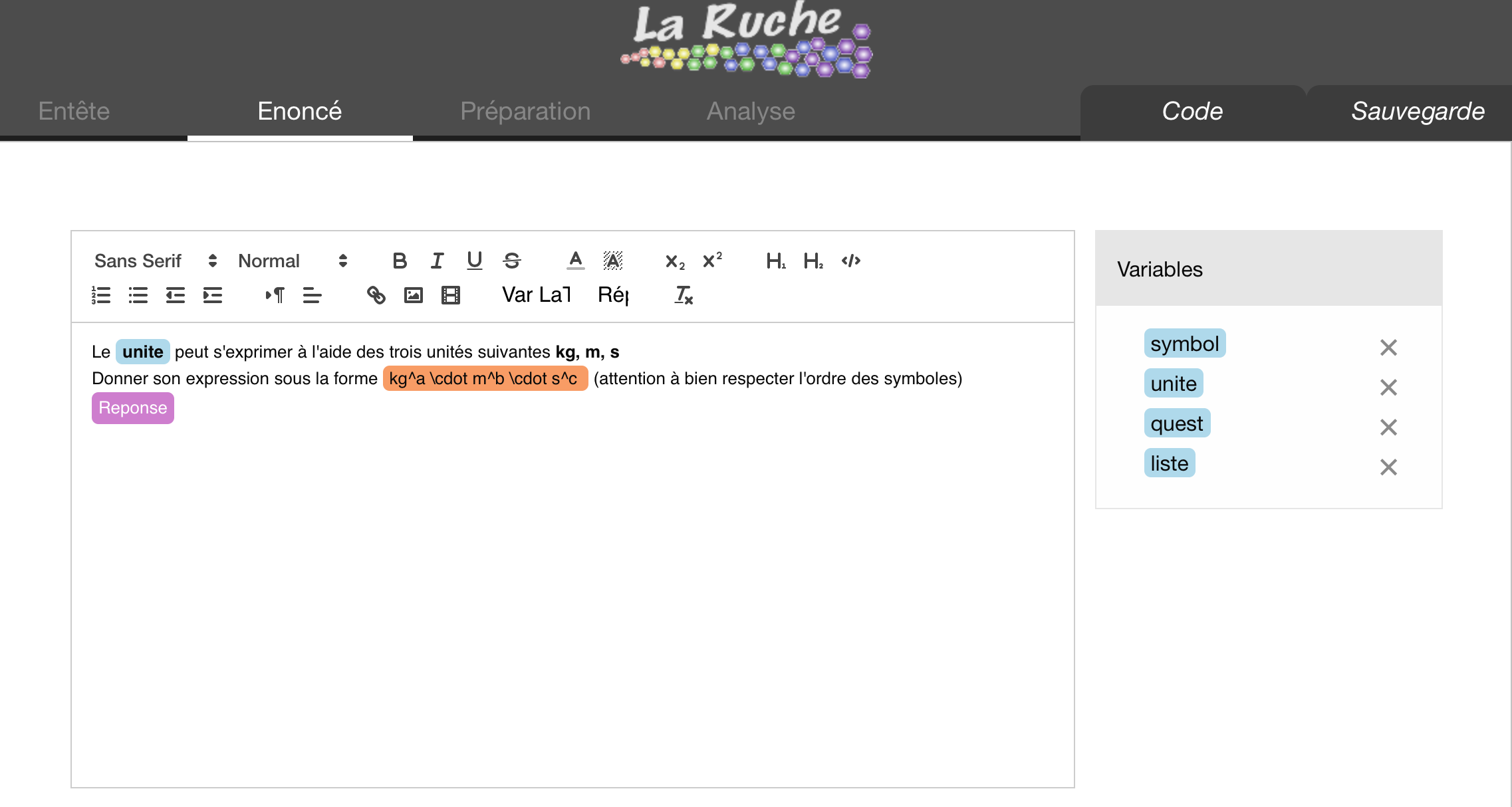
La Ruche, as an interactive editor of exercises, produces OEF (Open Exercise Format) code which is specific to WIMS. This code may be sent or copied to a WIMS server that will generate the exercise, show it to a student, mark it, etc... WIMS aiming at the development of exercises by teachers, it appeared that some people were not confident in writing code in the OEF language (you probably know some of them…). This is why we started the development of an editor that would be as much as possible interactive, wysiwyg and that would interact with a WIMS server.

The editor is a client side program written in JavaScript and using for now some open-source parts or packages :

* the [Foundation](http://foundation.zurb.com) framework
* the [Blockly](https://developers.google.com/blockly/) interactive programing editor
* the [Quill](http://quilljs.com) word editor which itself uses
* the [KaTex](https://khan.github.io/KaTeX/) library (mathematical equations display)

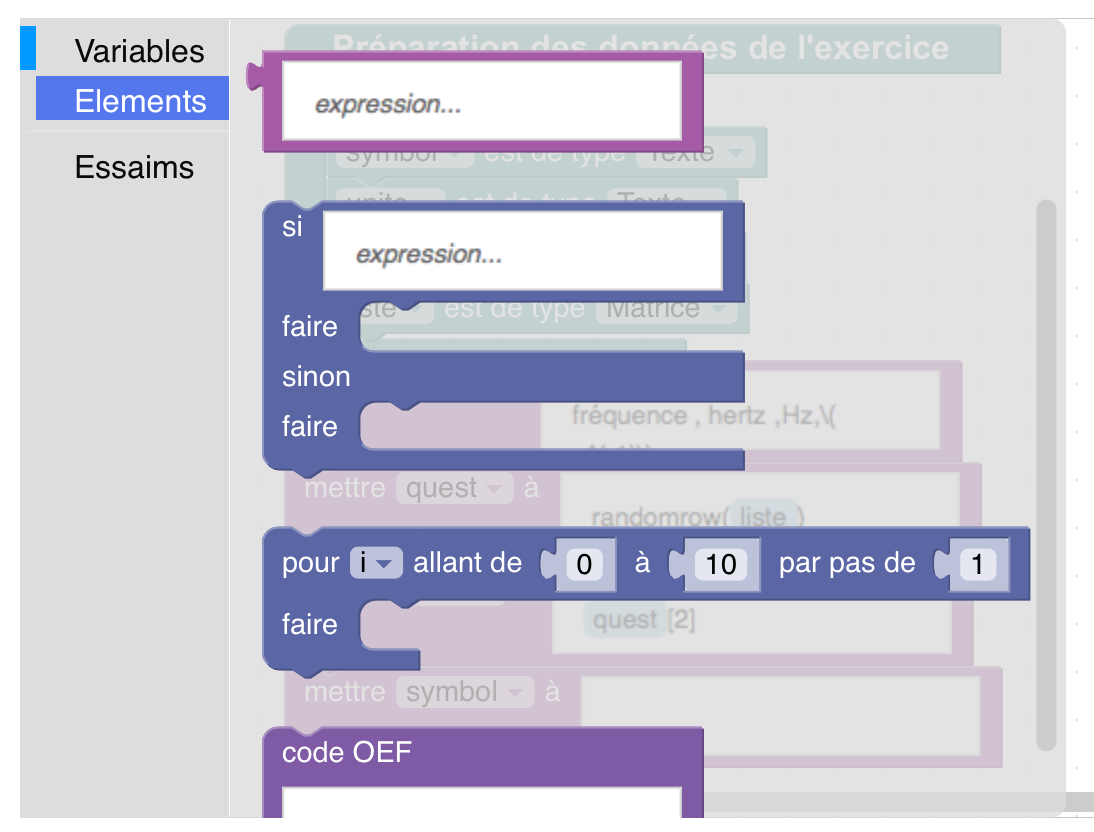
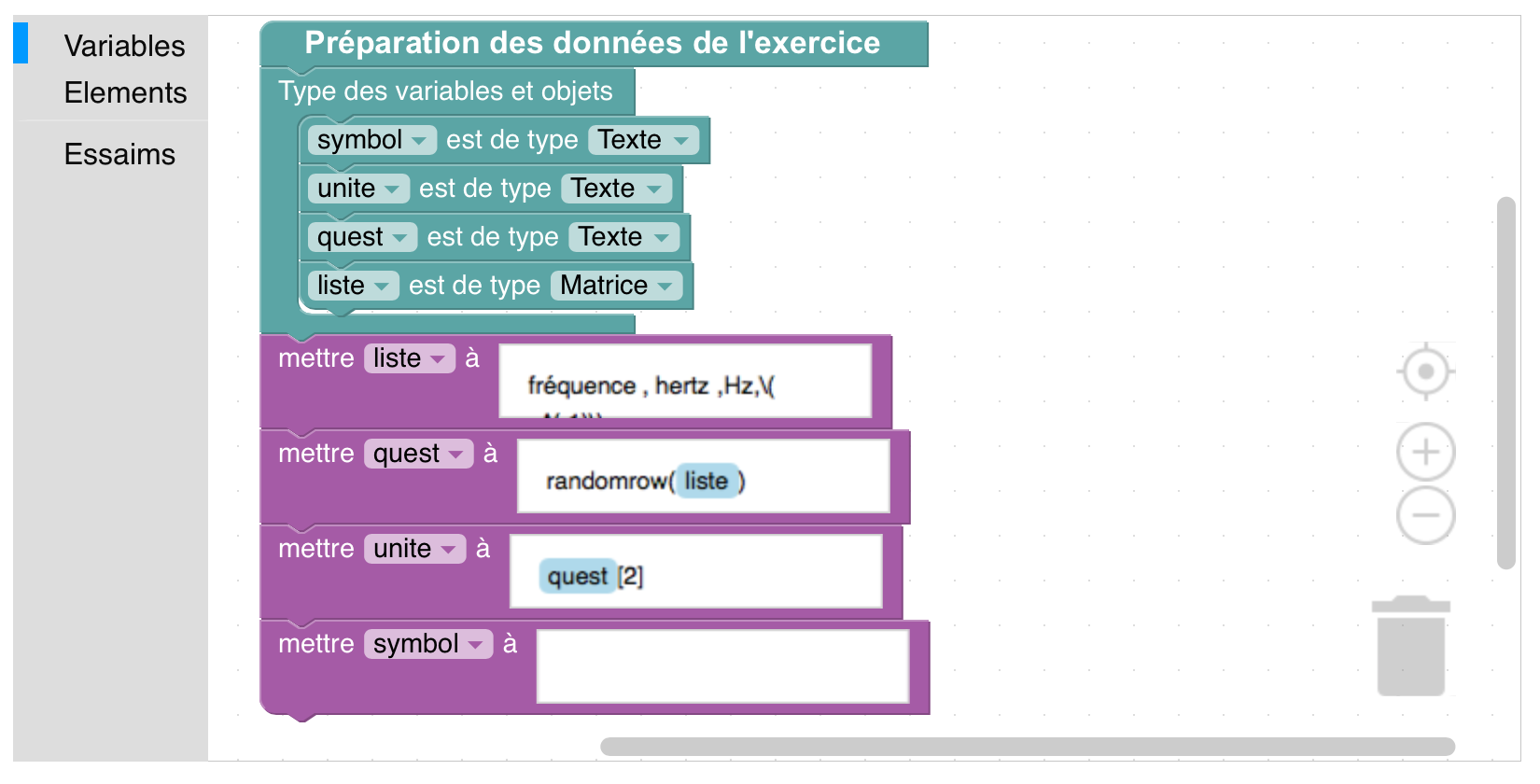
as well as somme utility packages and libraries :

* the JQuery library
* the [highlight](https://highlightjs.org/) library
* the [html2canvas](http://html2canvas.hertzen.com/) library



The editor is divided into four main and two auxiliary parts/tabs :

* the **Header** tab where the user enters meta-data for the exercise (title, author name, etc…)
* the **Statement** tab where the user enters the statement of his exercise in a Rich Text Editor, namely a special [Quill](http://quilljs.com) editor. This statement may contain references to variables, drawings, objects that are prepared in the **Preparation** tab
* the **Preparation** tab, where the user can make all the needed computations to set the variables, for example setting some of them as random. To that effect, a graphical Blockly editor is used.

* the **Analysis** tab, where the answer type for the student is set, as well as all necessary action (in another Blockly editor) after the student has entered his answer and pushed the "Send" button.

the two auxiliary tabs are :

* the **Code** tab where the OEF code is generated and displayed, so as to be copied to the WIMS server.
* the **Saving** tab that is used to generate a JSON format string corresponding to the exercise, save it or restore the exercise from a previously saved string.

## Constraints

For various reasons, we decided of the following constraints on the code development :

* the code is and should remain open-source
* the client-side is written in JavaScript and HTML5 (no Java)
* the code comments are in english
* the naming conventions will be discussed with the developpers
* vanilla JS is good, with JQuery where needed. The Foundation framework is good but should not be used very extensively so as not to prevent future evolutions (this may be negotiated on a case by case basis)
* the development is done with an Agile Software Development approach, to be discussed with the developers. This is needed because of the possible changes or adaptations that are inherent to the project.
* the development is done in collaboration with the WIMS and La Ruche developers

# Project details

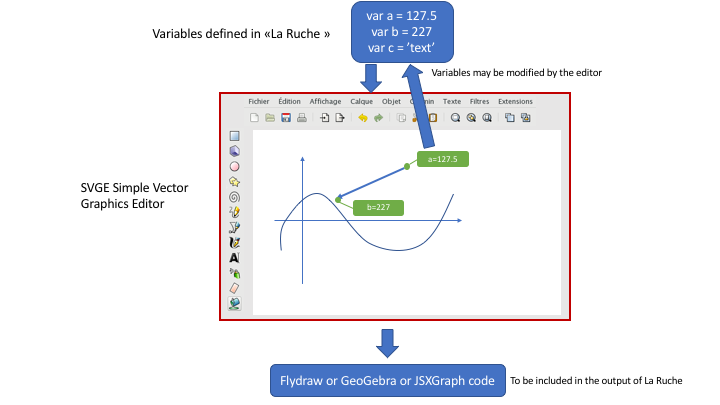
## Rationale

In order to create a drawing within a WIMS exercise, a user has to use a programming language called “Flydraw” that is included into WIMS. Our aim is to build a simple vector graphics editor (SVGE) for this language. Starting from an open source project like “[JSXGraph](https://jsxgraph.uni-bayreuth.de/wp/index.html)” or “[Paper.js](http://paperjs.org/)” or “[GeoGebra](https://www.geogebra.org/)”, we want to build a graphical interface that allows the user to build some vector graphics, make some links with variables that were defined elsewhere in the La Ruche editor and generate automatically the corresponding Flydraw code. Ultimately, the SVGE editor should be able to generate any code for a GeoGebra, JSXGraph or other library. Those libraries are already included into WIMS and used by some exercises.

The integration within the La Ruche editor has still to be discussed, whether it is a special div that is overlaid to the Blockly editor or part of a Blockly block.

## Principles

The principle of the vector drawing editor is described on the following figure :



The user defines a set of variables in the **Preparation** section of La Ruche. As this is a Blockly program editor, one of the blocks will be dedicated to drawings. When clicking on such a block, the SVGE should appear and the user may freely draw, some of the elements being tied to the values of the variables defined above.

When a variable is empty, its value is replaced by the value assigned in the editor and appears in a box. When a variable is not empty (may contain a text formula), the variable value appears in a box but it is not superseded by the current value in the editor.

The project has the following steps :

* + Development of a prototype with simple commands :
    - Line, Arrow, Polyline, Circle, Point, Rectangle, Text
    - Characteristics to be defined (grid ? sticky points ?...)
  + Tests and integration into LaRuche
  + Development of improvements and missing simple commands
    - Grid
    - Mathematical function drawing
    - External image inclusion
    - Management of graph evolution/variation, creation of mini-films
    - Saving, Restoring
    - …
  + Tests and integration into La Ruche
  + Development of more sophisticated functions
    - Library of pre-defined complex elements (axes, some 3D projected elements, etc…)
    - User-generated complex elements, saving them to the server and restoring them
    - Other functions that are dependent on the library used (Paper.js, JSXGraph or GeoGebra or…)
  + Tests and integration into La Ruche

The project extends to March 2019, there are a few user testing and feedback sessions foreseen (so called “WIMSATHONs”).