Project report

JAVA/POO/UML

Cesi Exia

A1-2018

* Group members:

MAITRE Maxime

AZZOUZI CLAUSEL Zacharia

BOULESTEIX Tristan

KARDOUS Jean-Pierre

Summary

[I. Context/Goals 0](#_Toc515913065)

[II. Projected schedule / Real schedule 1](#_Toc515913066)

[III. Implementation of the UML 1](#_Toc515913067)

[A. Diagrams 1](#_Toc515913068)

[1. Diagram of packages 1](#_Toc515913069)

[B. Codes 1](#_Toc515913070)

[1. Model 1](#_Toc515913071)

[2. View 1](#_Toc515913072)

[3. Controller 1](#_Toc515913073)

[4. Contract 1](#_Toc515913074)

[IV. Organization of the database 1](#_Toc515913075)

[A. Entity relationship diagram 1](#_Toc515913076)

[B. SQL 1](#_Toc515913077)

[V. Conclusion 1](#_Toc515913078)

[A. Conclusion of project 1](#_Toc515913079)

[B. Problems encountered 1](#_Toc515913080)

[C. Group report 1](#_Toc515913081)

[D. Personal report 1](#_Toc515913082)

# Context/Goals

The main goal of our project is to recreate the Lorann video game in Java language with five different levels. The levels are stored in a database.

|  |  |
| --- | --- |
|  | Lorann, the main character |
|  | The multicolored spell |
|  | Energy sphere |
|  | The exit door of the level |
|  | The impassable elements of scenery |
|  | The four demons |
|  | Treasures to earn points |

Figure 1- Different sprites used

The figure above shows the different sprites that are given to us and that must be used for our program. To kill demons, Lorann can launch a spell. He can only use his spell once, if he wants to resume his spell, he must touch it.

# Projected schedule / Real schedule

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Days  Name | Friday  05/25/18 | Saturday  05/26/18 | Sunday  05/27/18 | Monday  05/28/18 | Tuesday  05/29/18 | Wednesday  05/30/18 | Thursday  05/31/18 | Friday  06/01/18 | Saturday  06/02/18 | Sunday  06/03/18 | Monday  06/04/18 | Tuesday  06/05/18 |
| AZZOUZI Zacharia | Discovering the project subject and group organization | | | Reflection about the project structure and making the first UML | | Making the plugged code about the DAO, creating the database and maps | Finish the DAO and popup code to select the level | Starting Junit tests | Starting the Javadoc, continue Junit tests and modify README.md | | Trying to repair the Junit test, and upgrade the Javadoc | Finish deliverables and compilation of the program |
| BOULESTEIX  Tristan | Making the plugged code of model | Finish model code and add part in the model code | Update Controller code and making contract code | Starting the Javadoc, finishing the last part of the code and debugging | | Continue to debug the code (add new functionality) and try to repair Junit tests |
| MAITRE  Maxime | Making the plugged code of view | Finish code of the frame and update class diagram/making the component diagram | Making the Sequence diagram and update the class diagram | Starting the Javadoc, update UML | | Create the second sequence diagram and update UML |
| KARDOUS  Jean-Pierre | Making the plugged code of controller | Making the package diagram and start the project report | Making the sequence diagram and create the score popup | Starting the Javadoc, and update the report | | Create the second sequence diagram and update the project report |

# Implementation of the UML

## Diagrams

### Diagram of packages

To create the diagram of packages, we used the class diagram. In fact, we took the name of the different packages and we linked them together with dependency links.

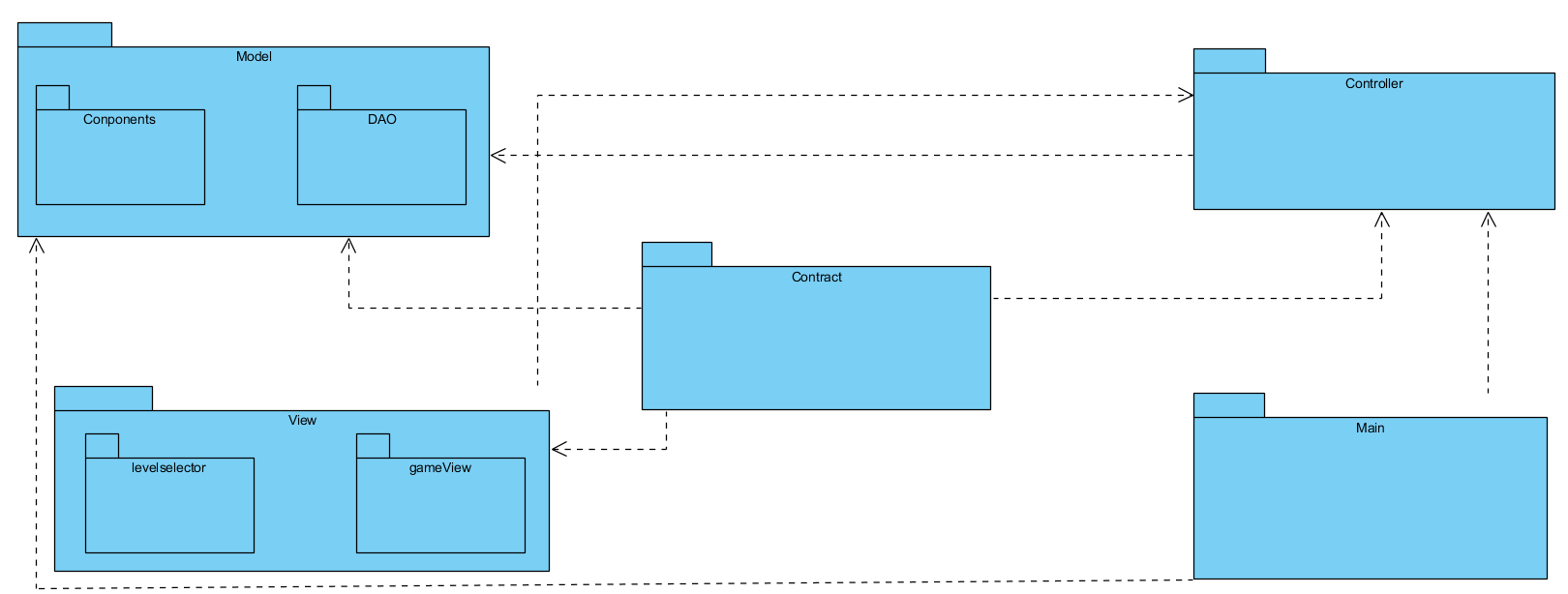


Figure 2- Diagram of Packages

## Codes

### Model

In the package Model, there are every component which will be used in the game.

Furthermore, there are also classes which can read external data like the database or sprites of the components

### View

In the package View, there is the code of the frame and class to update the display. It has also a list about the selection of different level.

### Controller

In the package Controller, there are all the element that make the link between the model and the view. It contains also alternate thread which run the game, and which move the components.

### Contract

In Contract, there are all interfaces which make link between all other packages.

# Organization of the database

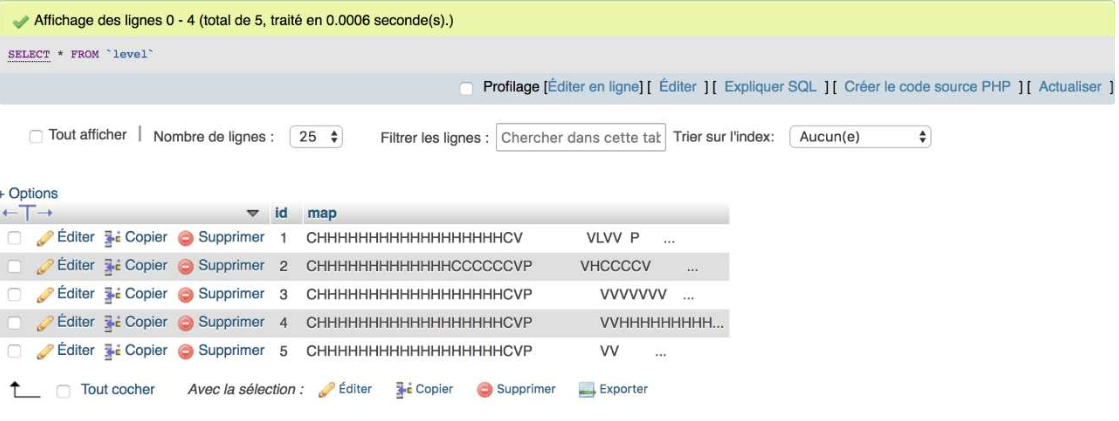
The database is used to store the different levels which are related with maps.

Figure 3- Database Table

Furthermore, we used stored procedure to find a map by it ID. We select the map we want to display. This map is stored in the table “level”.

## Entity relationship diagram

We have stored procedure which make link with all level which have an ID. So, we must create an entity relationship diagram which have two characteristics, an ID which have a unique number and the map which use a string.



Figure 4- Entity relationship diagram of project

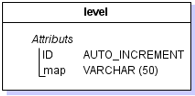


Figure 5- Logical model of data

## SQL

To have the possibility to find map by ID we have created a stored procedure which select the map from level with parameters the ID of the map.

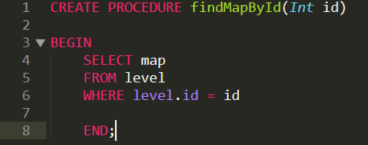


Figure 6- The SQL code of the project

# Conclusion

## Conclusion of project

## Problems encountered

During this project, even if it was easier than expected, we have meet some issues. The biggest problem was about Maven. The dependencies caused a lot of problems. For example, we could not generate a SureFire report with Maven despite the command “run as JUNIT Test” in Eclipse worked perfectly.

Another issue was with Git. One of our teammate was unable to sync the project with git desktop. He succeed by using git on Eclipse but he has to reconfigure the URL of the project each time because git seem to forgot the settings when he pull.

We also have some strange error like a null pointer exception which seem to appear randomly when we launch the spell but doesn’t append when we debug the program.

## Group report

## Personal report

**Tristan BOULESTEIX:**

I found this project really interesting and I really enjoyed working with this group. I already known Java before the teaching unity (UE) but I still learned a lot of things like the utility of the interfaces and the part Contract (thanks to Maven dependencies) and how to generate jxr, surefire report. I think this project doesn’t helped me to be a good programmer in Java, but it makes me learned to respect the rules, the Design Pattern…

**Jean-Pierre KARDOUS:**