

**An abandonware recreation**

***Java / OOP / UML Progamming***

**May / June 2018**

**Exia CESI Orleans – Year 1**

**Joël DIDIER, Nicolas DRAPIER, Bastien LE GALL, Thomas LOURENCO**

Table of Contents

[1. Introduction 5](#_Toc515881782)

[2. Game Manual 6](#_Toc515881783)

[2.1. Synopsis 6](#_Toc515881784)

[2.2. Background 7](#_Toc515881785)

[2.3. Non-implemented Features 7](#_Toc515881786)

[2.4. Installation 8](#_Toc515881787)

[2.4.1. Prerequisites 8](#_Toc515881788)

[2.4.2. Starting the game 8](#_Toc515881789)

[2.5. Game Description 8](#_Toc515881790)

[2.5.1. Level 8](#_Toc515881791)

[2.5.2. Spell 10](#_Toc515881792)

[2.5.3. Monsters 11](#_Toc515881793)

[2.5.4. Rules 11](#_Toc515881794)

[2.6. Game Controls 12](#_Toc515881795)

[2.6.1. Moving Lorann 12](#_Toc515881796)

[2.6.2. Cast the Spell 14](#_Toc515881797)

[3. Project Management 15](#_Toc515881799)

[3.1. Team Members 16](#_Toc515881800)

[3.2. Gantt Diagram 17](#_Toc515881801)

[4. Technical Part 18](#_Toc515881802)

[4.1. The Server 18](#_Toc515881803)

[4.2. The Database 18](#_Toc515881804)

[4.2.1. Basic Configuration 18](#_Toc515881805)

[4.2.2. Database Design 19](#_Toc515881806)

[4.2.3. Database Configuration 20](#_Toc515881807)

[4. UML Diagrams 22](#_Toc515881808)

[4.1. Classes Diagrams 23](#_Toc515881809)

[4.1.1. Model 23](#_Toc515881810)

[4.1.2. View 24](#_Toc515881811)

[4.1.3. Controller 25](#_Toc515881812)

[4.2. Packages Diagram 25](#_Toc515881813)

[4.3. Components Diagram 26](#_Toc515881814)

[5. Reports 27](#_Toc515881815)

[5.1. SureFire 27](#_Toc515881816)

[5.2. Javadoc 27](#_Toc515881817)

[5.3. JXR 27](#_Toc515881818)

[6. Group Report 28](#_Toc515881819)

[7. Conclusion 29](#_Toc515881820)

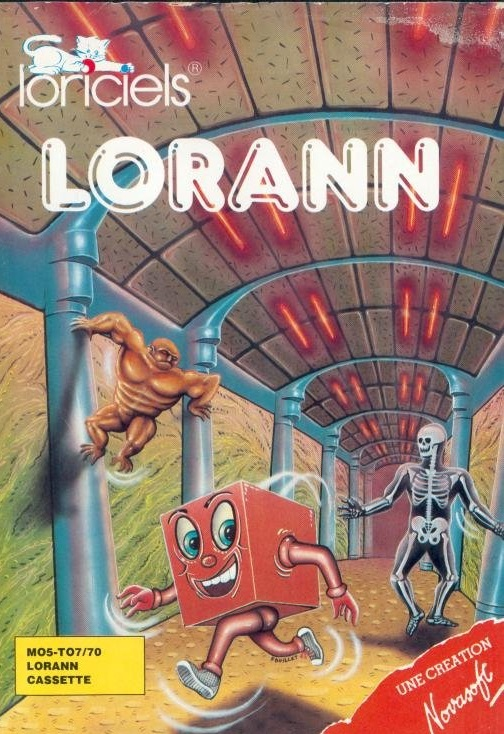
# Introduction

This document is a report on a Java Project achieved at Exia CESI.

This project consisted in recreating an abandonware (in this case: a game) entitled “Lorann”, only in Java.

You will find more information on this game and this project in the next pages.

This project was fully conceived in English as this was an important condition in the project success.



# Game Manual

## Synopsis

Lorann, Master of the Spell, the World of Nova-Ann needs your strength and your craftiness!

Free the hundred and one crypts detained by the Mask of Nekron, by reaching the Energy Bubble located in each.

Bring back the ephemeral Idols of the Life to the Cave, each will be worth two additional Lives.

Collect fabulous treasures which are waiting for you and your wealth will be tremendous!

Avoid the attacks of the four deadful Devils created by the vile Nekron, by cleverly using your very powerful and colorful Spell which obeys your will.

Glory is waiting of for you, Lorann… May the Spell protect you!

## Background

This game was created in 1985 by the former French Game Development Studio *Loriciels* and edited for Thomson MO5 and TO7 computers.

This is PacMan-like game which means that a single character moves square-by-square while collecting a maximum of items and defeat monsters to advance to the next stage.

## Non-implemented Features

Unlike the original game, the following features have not been implemented for timing reasons:

* All the 101 levels (only 5 levels have been implemented)
* The room allowing to choose the level (when a level is completed, the next one is automatically loaded)
* The ‘Sandbox’ Room allowing to create the player’s own levels

## Installation

### Prerequisites

* The Java Runtime Environment (JRE) version 8 and above must be installed to run the game
* An Internet Connection is required to run the game. Each level is stored in an online database. However, the player can set up his own database on his computer (MySQL Server 5.7 MAX), modify the required files in the project and recompile.

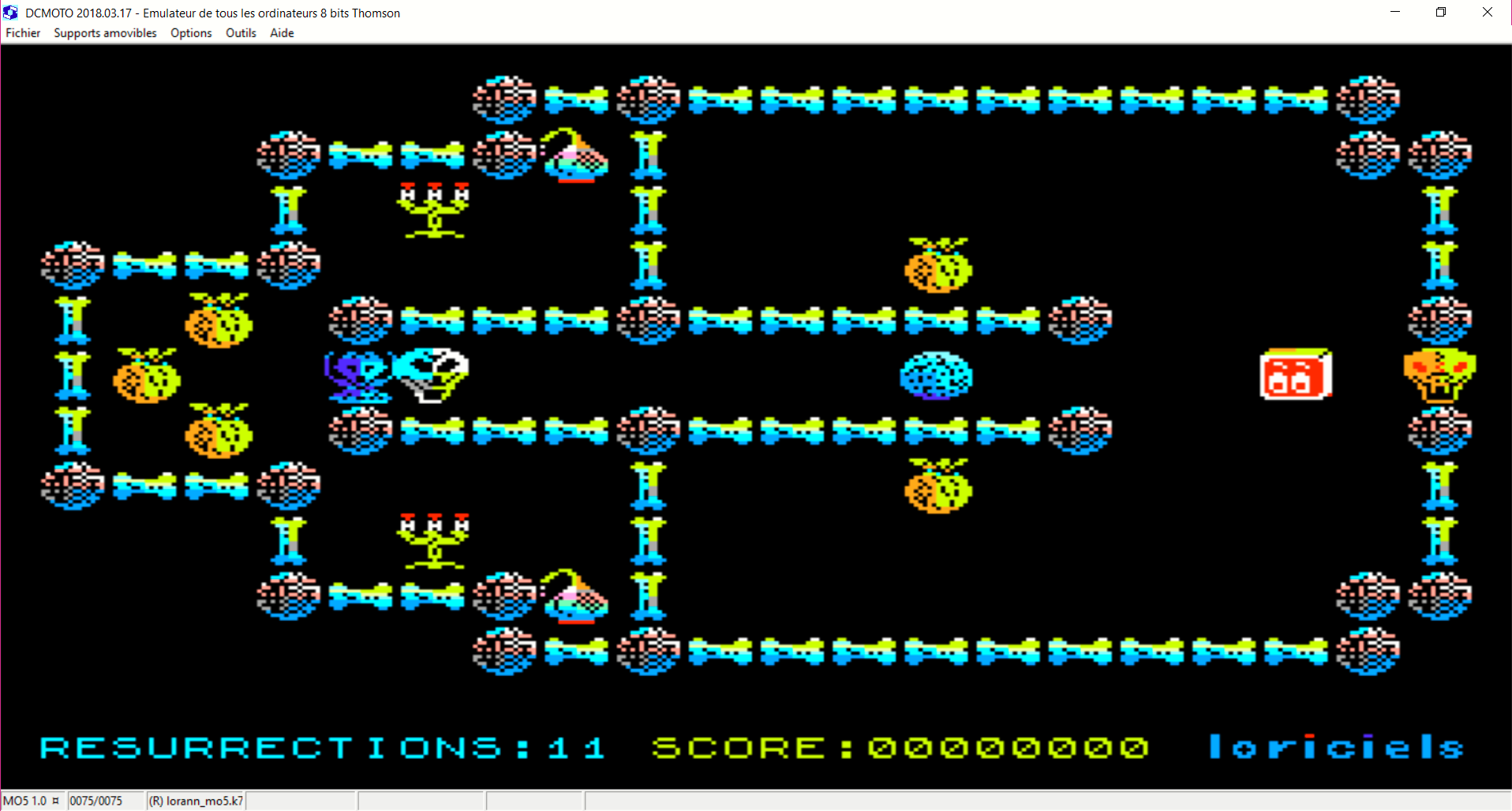
### Starting the game

The installation is simple since you must download the provided .JAR file.

When the .JAR is downloaded, double-click on it and the game will be launched automatically.

## Game Description

### Level



1. Screen Capture of the original game

The game is composed of 5 levels that you need to complete to finish the game.

A level contains different items and each of them have a specific design and behaviour. Here is a table that contains all the items that are in the game.

|  |  |  |
| --- | --- | --- |
| **Name** | **Sprite** | **Description** |
| Lorann | Une image contenant objet, trousse de secours  Description générée avec un niveau de confiance très élevé | Lorann is the hero of the game. You can control him in order to kill monsters, earn points and finish the level. |
| Spell | Une image contenant horloge, objet  Description générée avec un niveau de confiance très élevé Une image contenant horloge, objet  Description générée avec un niveau de confiance très élevé Une image contenant horloge, objet  Description générée avec un niveau de confiance très élevé Une image contenant horloge, objet, temps, monté  Description générée avec un niveau de confiance très élevé Une image contenant horloge, objet, jaune  Description générée avec un niveau de confiance très élevé | Spell is a powerful tool that can kill monsters. The spell can bounce on walls. |
| Purse |  | Purse is a 50 points bonus that can be picked up by Lorann. |
| Energy bubble | Une image contenant clipart  Description générée avec un niveau de confiance élevé | Energy bubble is the key of the level. This bubble must be picked up by Lorann in order to open the door and finish the level. |
| Wall |  | Walls defines the limits of the level. No one can pass through walls. |
| Door |  | Door is your way out of the level. But before trying to step out of the level you need to open the door by finding the Energy bubble. |
| Monster | Une image contenant objet  Description générée avec un niveau de confiance très élevé Une image contenant clipart  Description générée avec un niveau de confiance élevé | Monster is an NPC that will try to attack you. You can defeat monsters by using your spell wisely. |

2. Descriptive Table of the Level

**Goal**: The goal is to achieve to go out by the door. To open the door, you must find the Energy bubble.

**Points**: You can earn points in many ways. Collecting a Purse will grant you 50 points, killing a Monster will grant you 50 points and passing through the Door will grant you 100 points.

### Spell

Une image contenant horloge, objet

Description générée avec un niveau de confiance très élevé Une image contenant horloge, objet

Description générée avec un niveau de confiance très élevé Une image contenant horloge, objet

Description générée avec un niveau de confiance très élevé Une image contenant horloge, objet, temps, monté

Description générée avec un niveau de confiance très élevé Une image contenant horloge, objet, jaune

Description générée avec un niveau de confiance très élevé

3. Sprites for each color of the Spell

Lorann can cast a multicolour spell in order to kill a monster. The spell can bounce on walls and if it bounces back to Lorann, he will be able to cast the spell again. If the spell hits a monster, it kills him.

### Monsters

During your trip you will have to defeat monsters created by the vile Nekron, these creatures are called the four dreadful Devils. Each one of the can go in different ways such as **UP**, **DOWN**, **LEFT**, **RIGHT** and **DIAGONAL**. They will try to attack you so use your spell carefully!

|  |  |
| --- | --- |
| **Name** | **Sprite** |
| **(a.k.a Ghost)** | Une image contenant objet  Description générée avec un niveau de confiance très élevé |
| **(a.k.a Skull)** | Une image contenant clipart  Description générée avec un niveau de confiance élevé |
| **(a.k.a Totem)** |  |
| **(a.k.a Storm)** |  |

4. Descriptive Table of the Monsters

## Rules

This part of the manual centralizes the rules of the game.

* There is no time limit to finish any level
* Lorann can move in any direction (left, right, up, down, all diagonals)
* Lorann cannot go through walls (Pillars, Horizontal and Vertical Bones)
* Lorann can get killed if he encounters one of the followings:
  + The closed door leading to the next level
  + Any of the 4 monsters
* Lorann can kill the monsters only using his spell
* Lorann can cast his spell only once at a time, and must grab it back (by going on it’s way) to cast it again
* Monsters cannot go through walls

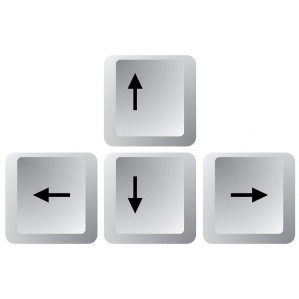
## Game Controls

### Moving Lorann

#### Any Keyboard Layout

To lead Lorann, you will have to use a few control keys.

On every keyboard, you can use the following Arrow keys:



**Move Up**

**Move Right**

**Move Left**

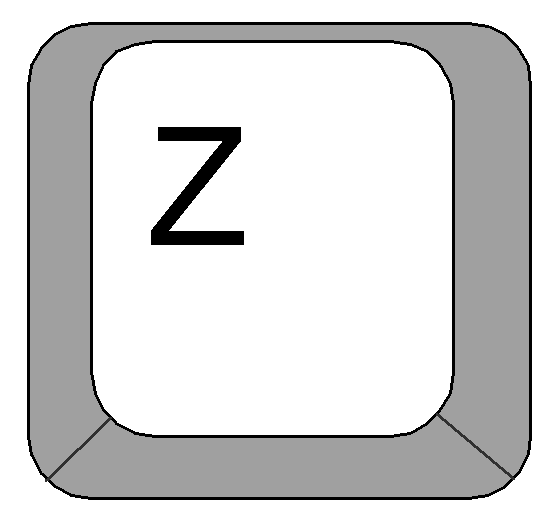
**Move Down**

#### AZERTY Keyboard

Along with the regular arrow keys, you can also use the following movement keys:

**Move Up**

**(Z)**



Une image contenant sport athlétique

Description générée avec un niveau de confiance élevé

**Move Down**

**(S)**

**Move Left**

**(Q)**

**Move Right**

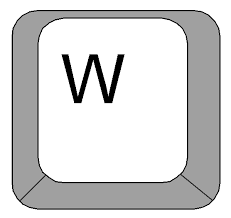
**(D)**

#### QWERTY Keyboard

Along with the regular arrow keys, you can also use the following movement keys:

**Move Up**

**(Z)**



Une image contenant sport athlétique

Description générée avec un niveau de confiance élevé

**Move Down**

**(S)**

**Move Left**

**(Q)**

**Move Right**

**(D)**

### Cast the Spell

Press the **SPACE** key to cast the Spell.

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

**Cast**

**(Space)**

# Project Management

## Team Members

Our group was composed of four persons:

* **Joël DIDIER**: *Project Manager, Database Master, Documentation Advisor, Level Designer*
* **Nicolas DRAPIER**: *Lead Java Developer*
* **Thomas LOURENCO**: *Documentation Master, Java Developer, Translator*
* **Bastien LE GALL**: *Java Developer, Game Tester*

# Technical Part

## The Server

The Project Manager and Database Master, Joël DIDIER, deployed a Cloud Server powered by Windows Server 2016 Datacenter.

The current IP Address of the server is: ***195.201.140.144***

This IP Address was written in the JDBC Function.

## The Database

### Basic Configuration

The server runs MySQL Community Server 5.7.22, installed using the MySQL Installer.

MySQL Server runs on port: ***3306***

(Be sure to allow port TCP 3306 in the Windows Firewall)

The following user has been set up:

**Username**: *player*

**Password:** *UnsecureLorannServer123*

This user, for security and privacy purposes, only has EXECUTE rights on the stored procedure.

### Database Design

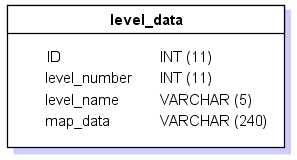
#### Database and Table naming

The database is named ‘***lorann***’ and the table containing the main data of each level is ‘***level\_data***’.

#### Database and Table Structure

The database is composed of only one table. It contains 4 columns (ID, level\_number, level\_name, map\_data).

The game is supposed to grab the content of map\_data on the condition that the provided level\_number is the same as the one in the database.



The database respects the First Normal Form because all fields are unique and cannot be devided in more columns.

#### Stored Procedure Structure

The Stored Procedure issues a SELECT … FROM … WHERE using a parameter that has been provided by the Java Program. This parameter is linked to the desired level.

The Store Procedure compares the parameter with the column containing the level number and sends the answer (the map data string) back to the game.

### Database Configuration

#### Database and Table Creation

***Deleting Database if it exists and create a brand new one.***

DROP DATABASE IF EXISTS lorann;

CREATE DATABASE lorann;

USE lorann;

***Creating Table ‘level\_data’.***

CREATE TABLE `level\_data` (

`id` int(11) NOT NULL,

`level\_number` int(11) NOT NULL,

`level\_name` varchar(5) NOT NULL,

`map\_data` varchar(255) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#### Stored Procedure Creation

***Creating Stored Procedure called ‘get\_map\_data’.***

DELIMITER //

CREATE PROCEDURE get\_map\_data(IN in\_level int)

BEGIN

SELECT map\_data

FROM level\_data

WHERE id = in\_level;

END //

DELIMITER ;

#### Filling database

***Filling all 5 levels with: ID, level number, level name and Map String Data.***

INSERT INTO `level\_data` (`id`, `level\_number`,`level\_name`, `map\_data`) VALUES

(1, 1, 'MAP\_1', 'AJJJJJJJJJJJJJJJJJAYXYYYYYYYYYYYYYYYYYYAXYAJJJJJJJJJJJJJAYYXXYYYYYYYYYYYYTYYYAYXXYAJJJJJJJJJJJAYYXYXXYXVYYYYYYYYYYYAYXYXXBXYYYYYYYYYYYYAYXYXXYAJJJJJJJJJJJAYYXYXXYWYYYSYYYYYYYYYYAYXXYAJJJJJJJJJJJJJAKYXAAYYYYYYYYYYYYYYYYYAYAJJJJJJJJJJJJJJJJAH'),

(2, 2, 'MAP\_2', 'AJJJJJJJJJJJJJJJJJJAXYYYUYYYYYYYYYYYYYYXXYYAJJJAYAJJJJJJAAYXXYAYYYYYYYYYYYYYXYYXXYXYAJJJJJJJJAYYXYHXXYXYXYYYYKYYYXYYXYAAXYXYXYYYYYYYYXYAASAAXYXYAJJJAYAJJAYYXYYXXYAYYYYYYYYYYYYBAYYXXYYAJJJJJJJJJJJJAAYXXYTYYYYYYYYYYYYYYYYXAJJJJJJJJJJJJJJJJAJA'),

(3, 3, 'MAP\_3', 'AJJJJJJJJJJJJJJJJJJAXYYYYYYYYKYYYYYYYYYXXYYYYYYYYYYYYYYYYYYXXYYYYYYYYWYYYYYYYYYXXYYYYYYYWBWYYYYYYYYXAAYAAYYYYWYYYYYAAYAAXYYYXYYYYYYYYYYXYYYXXYYYAJJJJJJJJJJAYYYXXYYYXYYYYYYYYYYXYYYXXYYYAYYVYYYYSYYAYHYXXYYYYYYYYYYYYYYYYYYXAJJJAJJJJJJJJJJAJJJA'),

(4, 4, 'MAP\_4', 'AJJJJAYYAJJAYYAJJJJAXYWYYXYYXYYXYYXYYYYXXWBWYAJJAYYAJJAYWYYXXYWYYYYYYYYYYYYYYYYXAJJJJAYYAJJJJJAYAJJAYYYYYXYYYYYYYYXYXYYYAJJJJAYYAJJJJJAYAJJAXVYYSYYYXYYYYYYYYYYXXYYYYAJJAYYAJJAYYYYXXYHYYXYYXYYXYYXYYKYXXTYYUXYYXYYXYYXYYYYXAJJJJAYYAJJAYYAJJJJA'),

(5, 5, 'MAP\_5', 'AJJJJJJJAYYYYAJJJJJAXYYYYYYYXYYYYXSYYYUXXYAJJJAYXYYYYXYYAYYXXYXYYYYYXYYYYXYYWYYXAYXYAJJJAJJJJAYYWYYXHYXYXYYYXYYKYXYAWBYXAYXYAYAYAYYYYXYYWYYXXYXYYYXYYYYYYXYYWYYXXYAJJJJJAJJJJAYYWYYXXYYYYYYYYYYYYYYYAYYXAYYYYYYYAJJJJATYYYVXAAJJJJJJAYYYYAJJJJJA');

# 4. UML Diagrams

## 4.1. Classes Diagrams

See in the deliverables folder: Deliverables/Java\_Program\_Diagrams

## 4.2. Packages Diagram

See in the deliverables folder: Deliverables/Java\_Program\_Diagrams

## 4.3. Components Diagram

See in the deliverables folder: Deliverables/Java\_Program\_Diagrams

# Reports

## SureFire

Because we are experiencing Circular Dependencies, we put all tests in the main package.

Thus, all tests are available in **Lorann/target/staging/main**

## Javadoc

They are available in **Lorann/target/staging/apidocs**

## Javadoc Tests

They are available in **Lorann/target/staging/testapidocs**

## JXR

They are available in **Lorann/target/staging/xref**

# Group Self-Assessment

Based on the provided assessment sheet, here is a table showing the progress in the project and the justification.

**Functional Evaluation**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Score** | **Explanation** |
| **Number of levels available and working** | **5** | **All five levels are stored in the DB and are successfully loaded.** |
| **Number of demons with different behaviors available and working** | **0.5** | **The Artificial Intelligence is present in the code as comment (in model / myModel : MOVABLEITEM\_Ennemy class) because the repaint (view refresh) is not properly working. But if it was working, then the AI would work as expected and the monsters would chase Lorann** |
| **The gate only opens when Lorann grabs the crystal ball** | **1** | **It works as expected in the game** |
| **Demons are chasing Lorann** | **0.25** | **See “Numbers of demons with different behaviors…”** |
| **Lorann fires one and only one multicolor ball** | **0.5** | **The ball is instantiated, but only in front of Lorann (his last direction). The ball does not follow Lorann, does not move, stays at the same place, but kills if a monster is on the same “square”.** |
| **Lorann can guide the multicolor ball once it has been fired** | **0** | **Not implemented.** |
| **Lorann is not killed by the multicolor ball** | **0.5** | **Lorann can go on the same square as the spell and does not die.** |
| **Lorann is killed by demons when there is a contact** | **1** | **Lorann is killed by the demons if it is on the same square as the demon. Even though the demon is not moving, the check is based on if Lorann is on the same square of the demon, so it should do the same if the demons were moving. Also, if a demon got in a square and Lorann goes on it just right after, then Lorann is killed because he went on the same square as the demon, at the “right” moment.** |
| **Lorann is able to move in any direction, including diagonals** | **3** | **Working as in the demo** |
| **Lorann can grab the purses or the crystal ball** | **1** | **Working as in the demo** |
| **The multicolor ball cannot go through blocking elements** | **0.5** | **The multicolor ball cannot be instantiated on any wall as shown in the demo (otherwise, the square where the wall is located would go black, which wasn’t the case).** |
| **Lorann cannot go through blocking elements** | **1** | **Working as in the demo. Note: Lorann also dies when he hits the closed door.** |
| **Monsters cannot go through blocking elements** | **0.5** | **As shown in the JUnit test (CollisionTest), a blocking element does not return the ID “3” which says the square is empty, as an enemy can only go on empty squares or on Lorann (and kill him). Check the getColliderMonster method in MOVABLEITEM\_Ennemy and the myIA method just under it to see the implementation.** |
| **Lorann is animated when he doesn’t move** | **1** | **Working as in the demo** |
| **The multicolor ball has its color changed when it moves** | **0.25** | **The multicolor ball animation is not repainted (repaint problem) and does not work because of it. If repaint was fixed, then it would be animated.** |
| **The presentation was only done in english** | **3** | **Only in English as heard/viewed** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code Evaluation** | **B** | **!** | **Max** | **Explanation** |
| **The code matches the design diagrams** | **2** | **3** | **2** | **Verified it and everything is matching the code/diagrams** |
| **No error when compiling** | **?** | **3** | **1** | **There are circular dependencies that, by default, lead Eclipse to throw an error. Else, if they are set as “Warning”, there is no error.** |
| **No warning when compiling** | **?** | **3** | **1** | **See above** |
| **The use of setters/getters is “automatic”** | **0.5** | **2** | **1** | **Most of the time, setters and getters are used. But sometimes they are not used**  **(done in a “speedy” way as we were short in time)** |
| **The use of abstract classes for code factorization is pertinent** | **0.5** | **2** | **2** | **We used abstract classes to do this but not for everything** |
| **The code of each class is short (max 100 lines)** | **1** | **2** | **2** | **We have some classes with way more than 100 lines but most of them are below 100 lines. Also, in some code, we left useless spaces taking some lines.** |
| **The code of each method is short**  **(max 15 lines)** | **1** | **2** | **2** | **Same as above (but with 15 lines).** |
| **La JavaDoc est présente pour toutes les méthodes et tous les attributs** | **1** | **2** | **1** | **It has been set for everything** |
| **The Javadoc was manually enhanced/customized and is not only an auto-generated code by JavaAutoDoc-like tools** | **2** | **2** | **2** | **Added more info on a lot of classes, methods and attributes** |
| **The code is in English** | **3** | **2** | **3** | **Not a single word in French (we believe it strongly)** |

**Code Evaluation**

**Deliverables Presence Evaluation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Deliverables presence** | **A** | **!** | **Max** | **Explanation** |
| **Maven was used** | **1** | **1** | **1** | **Always used, multiple compilation and packages, + see commits** |
| **Javadoc report is present** | **1** | **1** | **1** | **Here and alive** |
| **Tests Javadoc is present** | **1** | **1** | **1** | **Here and alive** |
| **JXR report is present** | **1** | **1** | **1** | **Here and alive** |
| **Surefire report is present** | **1** | **1** | **1** | **Here and alive** |
| **Components Diagram is present** | **1** | **1** | **1** | **Here and alive** |
| **Packages Diagram is present** | **1** | **1** | **1** | **Here and alive** |
| **Classes diagrams are present** | **1** | **1** | **1** | **Here and alive** |
| **Git was used during all the project (its not a repository where everything was uploaded once the project is over)** | **1** | **1** | **1** | **Yes. See commits (100+ commits)** |
| **All members of the team used the Git repository** | **1** | **1** | **1** | **Yes. See commits** |

**UML Evaluation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Design Evaluation** | **B** | **!** | **Max** | **Explanation** |
| **MVC pattern is always followed** | **2** | **3** | **2** | **As shown** |
| **No circular dependencies in packages in the same module** | **0** | **3** | **1** | **If we’re talking about circular dependencies between model, view, controller, then yes, there are.** |
| **Classes are regrouped in the sale subpackages per function domain.** | **0** | **2** | **2** | **Not regrouped.** |
| **Des interfaces sont utilisées lors de l'échange de données entre composants** | **2** | **3** | **2** | **Yes, they are used** |
| **Compositions are rare, in favor of agregations.** | **0** | **2** | **2** | **No (apparent) composition but not many agregations.** |
| **Le diagramme de composants est correct dans le formalisme** | **1** | **2** | **1** | **Respects formalism.** |
| **Le diagramme de packages est correct dans le formalisme** | **1** | **2** | **1** | **Respects formalism.** |
| **Les diagrammes de classes sont corrects dans le formalisme** | **1** | **3** | **1** | **Respects formalism.** |
| **The database respects the normal form** | **1** | **3** | **1** | **See in the Lorann Report for expalanation or DB script in Database (each field are unique and cannot be divided)** |
| **La base de données intègre un stockage pertinent des niveaux** | **2** | **2** | **2** | **The level storage is pertinent** |
| **Strategy DP was used in an appropriate way** | **1** | **2** | **2** | **Not often used.** |
| **The conception allows a minimal instanciation of each sprite** | **2** | **2** | **2** | **Sprites are loaded once and only the concerned ones are changed, not all the map.** |
| **Observer DP is present to refresh the view** | **1** | **3** | **1** |  |
| **Elements encapsulation in packages is optimized** | **2** | **1** | **2** | **We used the appropriate modifiers for the elements when possible** |
| **All diagrams are in English** | **2** | **2** | **2** | **They are (and we strongly believe it)** |
| **Another DP was wisely used** | **1** | **1** | **2** | **Factory DP was used** |
| **Lorann, Monsters and spell move are done in the same loop.** | **0** | **3** | **1** | **Lorann is directly moved with the arrows. So this is not working.** |

**Test Report and Javadoc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **JUnit Tests Evaluation** | **B** | **!** | **Max** | **Explanation** |
| Tests are present in View | 2 | 3 | 2 | We have these |
| Tests are present in Controller | 2 | 3 | 2 | We have these |
| Tests are present in Model | 2 | 3 | 2 | We have these |
| **Tests verifying exceptions are implemented** | **2** | **2** | **2** | **We have these, for example on the string gathered from the server.** |
| Tests on object collections are done | 0 | 2 | 1 | Not done |
| Tests on the right understanding and key pressure are implemented | 2 | 2 | 2 | We have these |
| Tests on the denying of limit values on setters/getters are used | 2 | 2 | 2 | We have these. We test the string gathered from the server |
| Tests on all public elements are done | 1 | 3 | 2 | Not on all, but still. |
| **The methods etUp() and setUpBeforeClass() are correctly used** | **0** | **1** | **1** | **Not used** |
| Tests have been factorized on general classes | 0 | 1 | 1 | Not done |