Lorann-Ex

Baptiste Saclier Maire Chiaverini Clément Chabrier Maxime Zupka

Mercredi 23 Juin 2016

Contents

Chapter 1

Planning

At the beginning of the project we had to organize all the project to give differents tasks to the members of the team.

1.1 Team

The team is composed of four persons each have a specific task in the project.

Baptiste Saclier *Project leader* : In charge of the organisation of the project and the controller of the software

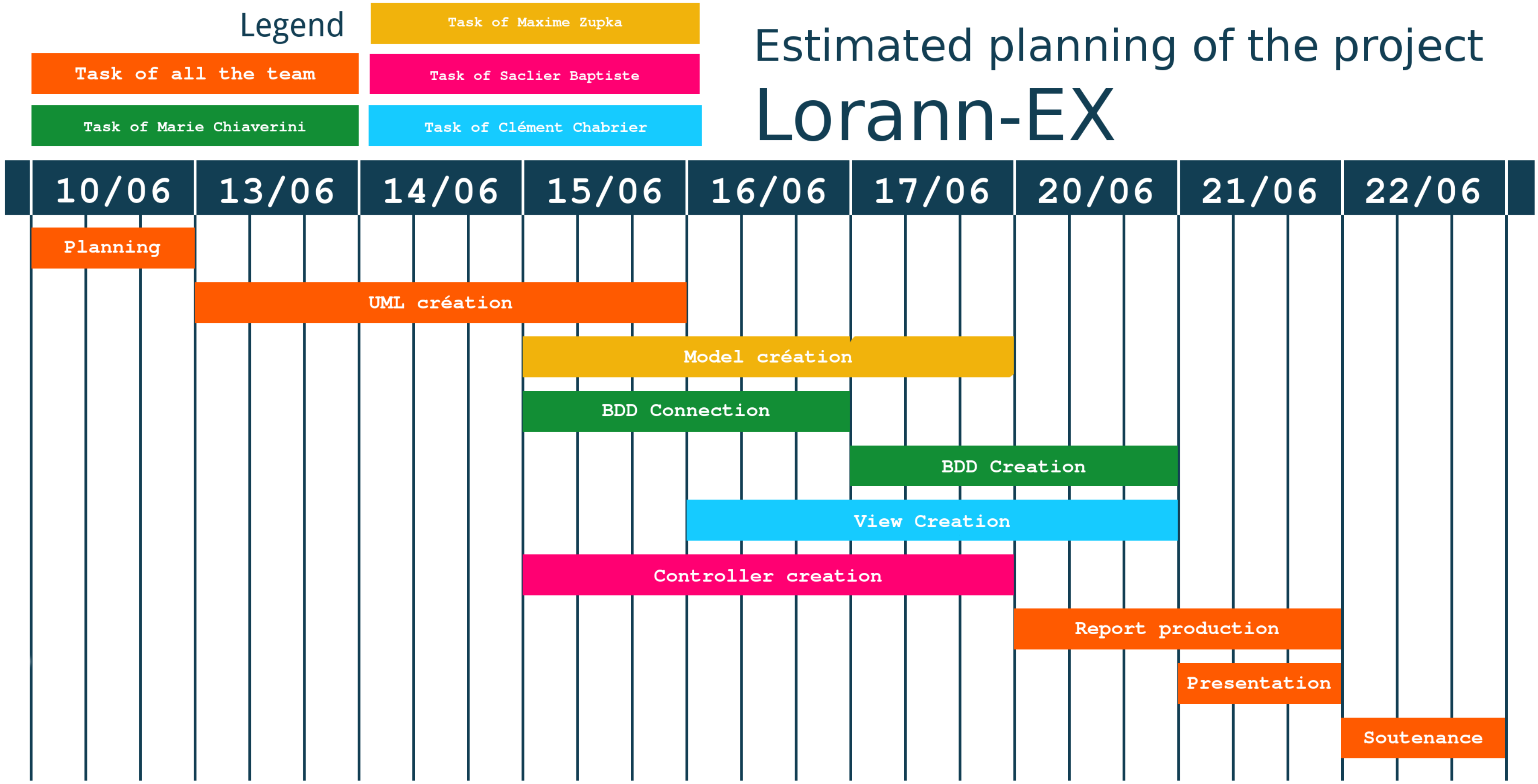
Marie Chiaverini In charge of the model of the game, loading systems and data storage

Clément Chabrier In charge of the view of the project, UML redactor and documentation master

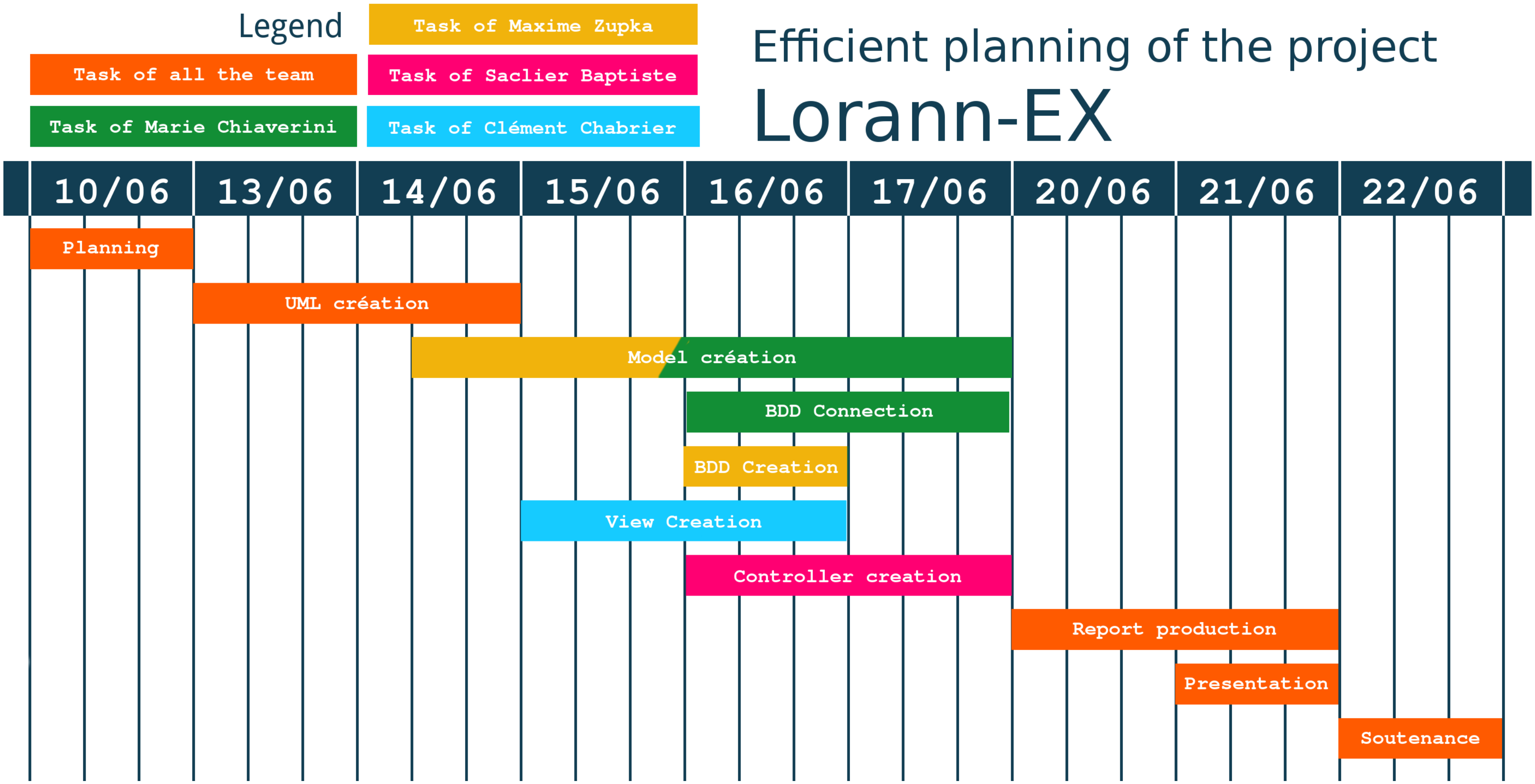
Maxime Zupka In charge of the design of the level, the documentation and the administration of the database.

1.2 Planning diagram

1.2.1 Estimated planning



1.2.2 Efficient planning



s

Chapter 2

Game manual

Welcome to the world of Lorann. In this dungeon you will discover a whole universe of danger in the maze of Nova-Ann. You have to finish all the levels to win the game.

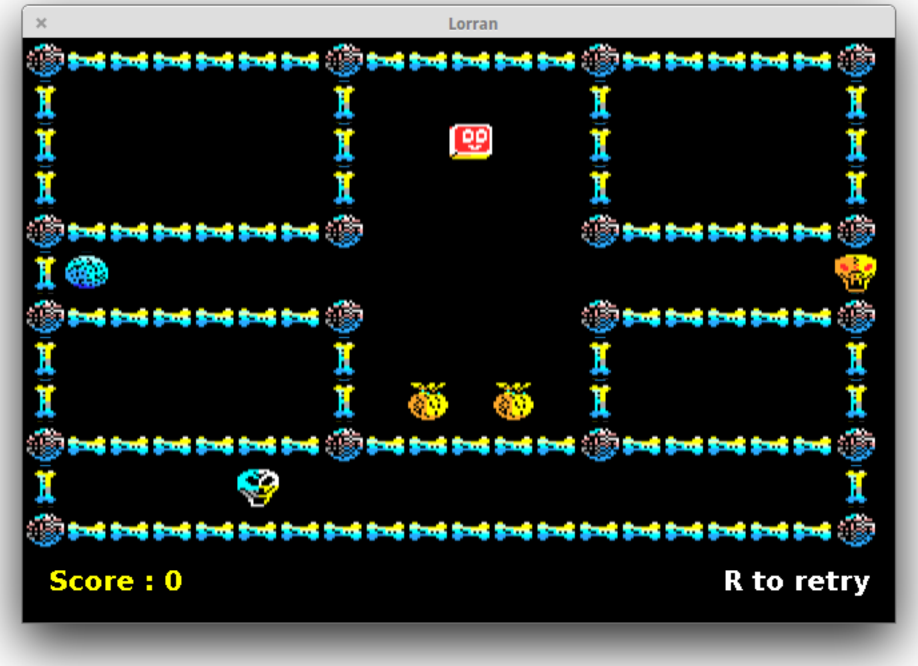
Good luck Lorann.

2.1 Installation

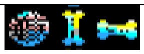
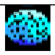
The installation of the game is pretty easy. Just download the executable *JAR* from *github.com* at the address `goo.gl/Mf4zIH`. And play. The database is distant and fully configurated.

2.2 Level

The first level of the game



A level contains many elements and each of them have a specific behavior. Next, you will find the components of this level and a short description of them.

Name	Sprite	Description
Lorann		The hero of the game. You can control him to finish the level, kill monsters and earn points.
Wall		The limits of the level. You and the monsters cannot pass trought this elements.
Purse		A 100 points bonus when lorann come from this item.
Magic Bubble		The key of the level. You need to get it to open the door and finish the level.
Door		On the left, the door of the level closed. It will kill you if you step on it. On the right, the door opened. Yous can finish the level if you step on it.
Monster		The monster is a deadly element. You cannot step on it but you can shoot it by launching a spell. There are 4 types of monster specified in the section 2.3

Goal The goal of the level is to go out by the opened door. Opened by the Magic bubble you took just before.

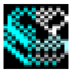
Points There are many possibilities to get points. First by taking money purses. By finishing the level, by taking the Magic bubble or by shooting monsters.

2.3 Monsters

There are four types of monsters. Each type have his own sprite and his own comportment. But they can all be killed by the spell of lorann and all killes lorann when he meet one of them.

Straight monster

This type of monster just go in a direction UP, DOWN, LEFT or RIGHT and bounce on walls. By killing it you will earn 150 points.



Diagonal monster

This type of monster go in a diagonal direction and bouce on the wall. If you kill it you will earn 225 points.

Random monster

This type of monster go in a random direction at any time. This random monster take a new direction on each tick. If you kill it you will earn 285 points

Following monster

This type of monster is the most dangerous. It will be follow Lorann in a straight line without avoid walls. If you kill it you will earn 315 points.

2.4 Controls

You have to control the position of lorann to go at the end of the level. To control him just use the arrow keys.

↑

← ↓ →

You can allso use the standards french or american FPS keys like.

↑
Z
← Q S D →
↓

↑
W
← Q S D →
↓

2.5 Spell

Lorann can cast a multicolor spell on the level to kill monsters by using the space key. It will be lauch in front of lorann and will bounce on the walls.

If the spell hits Lorann he will be able to cast an other one. If it hits a monster, he will die and the spell continue his path.

2.6 Retry

If Lorann die you can restart the level from the beginning by hitting the key R. Il will cost you 350 points and the level will be reloaded.

Chapter 3

Software's system

3.1 MVC

The MVC (model-view-controller), is a data pattern used in JAVA to build efficiently softwares. It has the property of separate the program in three categories.

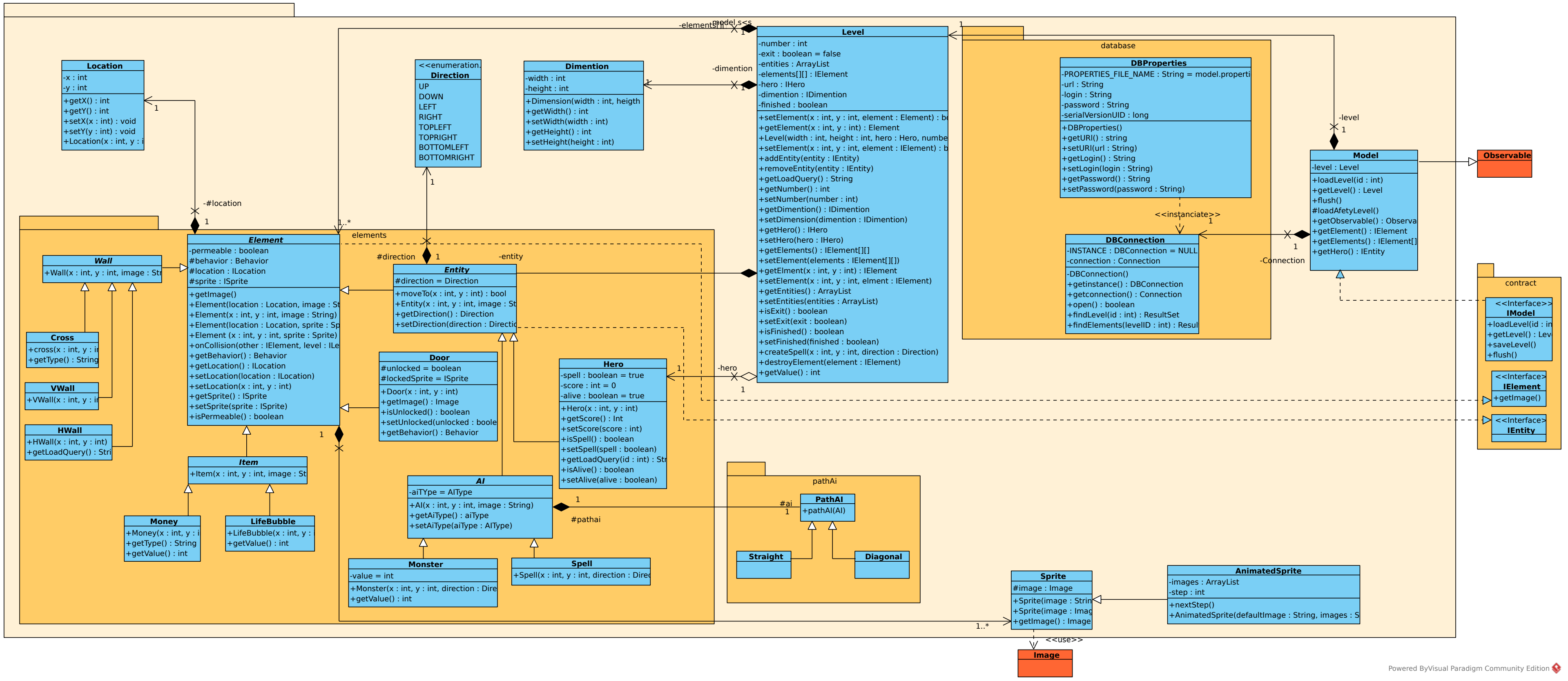
3.1.1 Model

The model part represents the software's core, it shall treat the data and tell to the view what it should show to the user.

In our case the model has multiple functions:

- To get data from the database
  - DBProperties class
  - DBConnection class
  - Model class
- To build the level to play, set the elements to their location and define their properties
  - Element class and all its derivatives
  - Location class
  - Level class
  - Dimention class
  - Direction enumeration
- To select what sprite will appear on screen
  - Sprite
  - AnimatedSprite

to see a graphic view of the model unit please report to the next page.


Location
-x : int
-y : int
+getX() : int
+getY() : int
+setX(x : int) : void
+setY(y : int) : void
+Location(x : int, y : i
1
-#location
1
<<enumeration.
Direction
UP
DOWN
LEFT
RIGHT
TOPLEFT
TOPRIGHT
BOTTOMLEFT
BOTTOMRIGHT
1
Dimention
-width : int
-height : int
+Dimension(width : int, heigth
+getWidth() : int
+setWidth(width : int)
+getHeight() : int
+setHeight(height : int)
1
-dimention
1
-elements[][]
model.s<s
1
Level
-number : int
-exit : boolean = false
-entities : ArrayList
-elements[][] : IElement
-hero : IHero
-dimention : IDimention
-finished : boolean
+setElement(x : int, y : int, element : Element) : bo
+getElement(x : int, y : int) : Element
+Level(width : int, height : int, hero : Hero, numbe
+setElement(x : int, y : int, element : IElement) : b
+addEntity(entity : IEntity)
+removeEntity(entity : IEntity)
+getLoadQuery() : String
+getNumber() : int
+setNumber(number : int)
+getDimention() : IDimention
+setDimension(dimention : IDimention)
+getHero() : IHero
+setHero(hero : IHero)
+getElements() : IElement[][]
+setElement(elements : IElement[][])
+getElment(x : int, y : int) : IElement
+setElement(x : int, y : int, elment : IElement)
+getEntities() : ArrayList
+setEntities(entities : ArrayList)
+isExit() : boolean
+setExit(exit : boolean)
+isFinished() : boolean
+setFinished(finished : boolean)
+createSpell(x : int, y : int, direction : Direction)
+destroyElement(element : IElement)
+getValue() : int
1
database
DBProperties
-PROPERTIES_FILE_NAME : String = model.properti
-url : String
-login : String
-password : String
-serialVersionUID : long
+DBProperties()
+getURI() : string
+setURI(url : String)
+getLogin() : String
+setLogin(login : String)
+getPassword() : String
+setPassword(password : String)
<<instanciate>>
1
DBConnection
-INSTANCE : DBConnection = NULL
-connection : Connection
-DBConnection()
+getinstance() : DBConnection
+getconnection() : Connection
+open() : boolean
+findLevel(id : int) : ResultSet
+findElements(levelID : int) : Resul
1
-Connection
-level
1
Model
-level : Level
+loadLevel(id : int)
+getLevel() : Level
+flush()
#loadAfetyLevel()
+getObservable() : Observa
+getElement() : IElement
+getElements() : IElement[]
+getHero() : IEntity
Observable
contract
<<Interface>>
IModel
+loadLevel(id : in
+getLevel() : Lev
+saveLevel()
+flush()
<<Interface>
IElement
+getImage()
<<Interface>
IEntity
1..*
elements
Element
-permeable : boolean
#behavior : Behavior
#location : ILocation
#sprite : ISprite
+getImage()
+Element(location : Location, image : St
+Element(x : int, y : int, image : String)
+Element(location : Location, sprite : Sp
+Element (x : int, y : int, sprite : Sprite)
+onCollision(other : IElement, level : ILe
+getBehavior() : Behavior
+getLocation() : ILocation
+setLocation(location : ILocation)
+setLocation(x : int, y : int)
+getSprite() : ISprite
+setSprite(sprite : ISprite)
+isPermeable() : boolean
1
Wall
+Wall(x : int, y : int, image : Str
Cross
+cross(x : int, y : i
+getType() : String
VWall
+VWall(x : int, y : i
HWall
+HWall(x : int, y : int)
+getLoadQuery() : Stri
Item
+Item(x : int, y : int, image : St
Money
+Money(x : int, y : i
+getType() : String
+getValue() : int
LifeBubble
+LifeBubble(x : int, y :
+getValue() : int
#direction
1
-entity
Entity
#direction = Direction
+moveTo(x : int, y : int) : bool
+Entity(x : int, y : int, image : St
+getDirection() : Direction
+setDirection(direction : Directio
Door
#unlocked = boolean
#lockedSprite = ISprite
+Door(x : int, y : int)
+getImage() : Image
+isUnlocked() : boolean
+setUnlocked(unlocked : boole
+getBehavior() : Behavior
Hero
-spell : boolean = true
-score : int = 0
-alive : boolean = true
+Hero(x : int, y : int)
+getScore() : Int
+setScore(score : int)
+isSpell() : boolean
+setSpell(spell : boolean)
+getLoadQuery(id : int) : Str
+isAlive() : boolean
+setAlive(alive : boolean)
1
-hero
1
AI
-aiTYpe = AIType
+AI(x : int, y : int, image : String)
+getAiType() : aiType
+setAiType(aiType : AIType)
1
#pathai
Monster
-value = int
+Monster(x : int, y : int, direction : Dire
+getValue() : int
Spell
+Spell(x : int, y : int, direction : Direc
pathAi
#ai
1
PathAI
+pathAI(AI)
Straight
Diagonal
Sprite
#image : Image
+Sprite(image : Strin
+Sprite(image : Imag
+getImage() : Image
1..*
AnimatedSprite
-images : ArrayList
-step : int
+nextStep()
+AnimatedSprite(defaultImage : String, images : S
<<use>>
Image
Powered ByVisual Paradigm Community Edition

3.1.2 View

The view is the visual interface trough which one the user shall "see" the program.

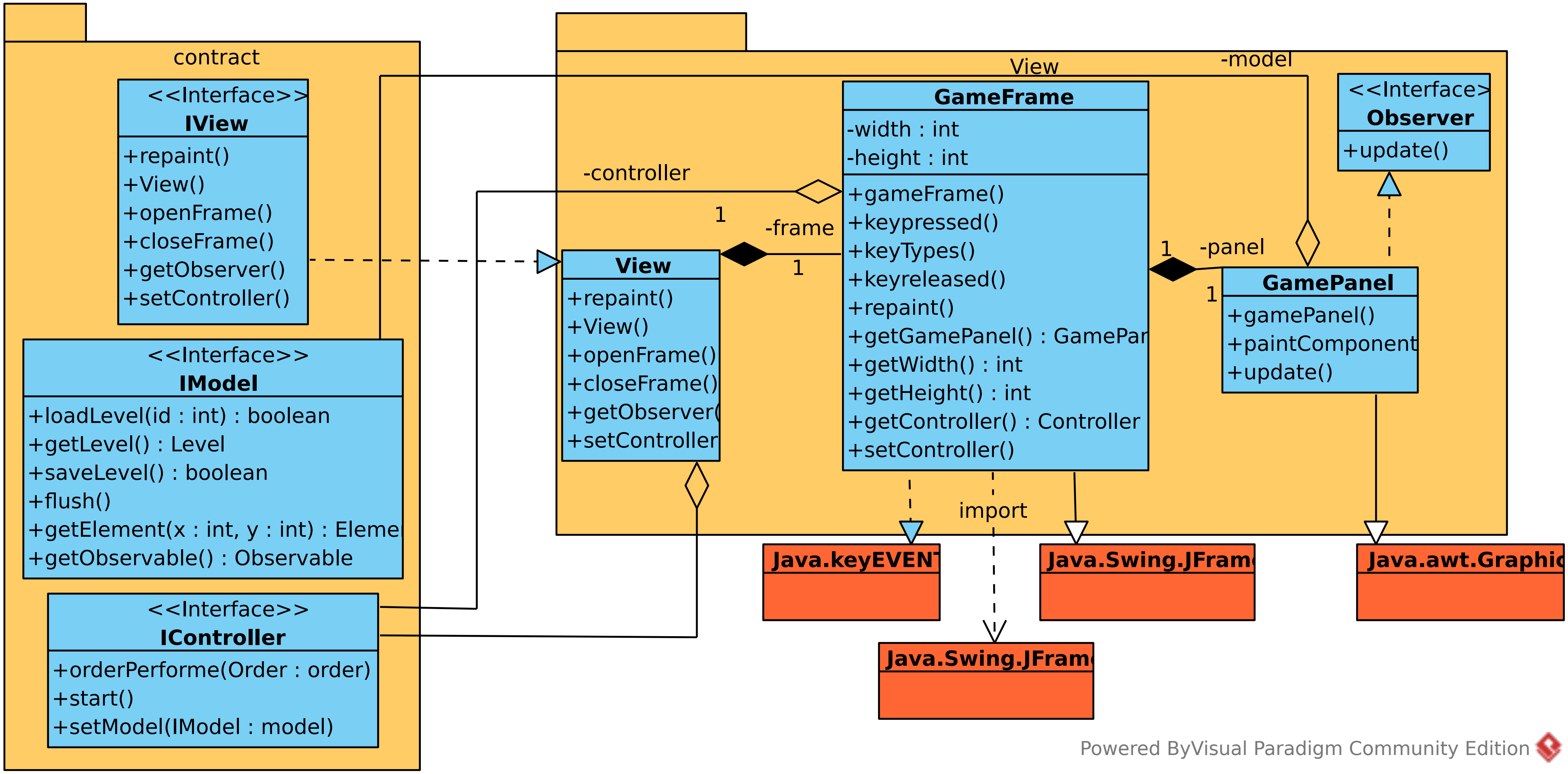
It has to show the results of the calculations from the model unit and to "listen" every action from the user that may cause a change in the program (mouse click, key pressed etc. . .)

In our game the view part:

- Open a frame to see the game
  - GameFrame class
- Display graphic elements on the screen
  - GamePanel class
- Update what the user see, based on the model's information
  - View class
- Keyes listener
  - GameFrame class

About the model and the view In order to inform the view unit about the model changments, we have to set up a pattern observer. We create an observer interface that will notice the view every possible changes and will make it react, so the information on screen update in real time.

to see a graphic view of the view unit please report to the next page.


contract
<<Interface>>
IView
+repaint()
+View()
+openFrame()
+closeFrame()
+getObserver()
+setController()
<<Interface>>
IModel
+loadLevel(id : int) : boolean
+getLevel() : Level
+saveLevel() : boolean
+flush()
+getElement(x : int, y : int) : Eleme
+getObservable() : Observable
<<Interface>>
IController
+orderPerforme(Order : order)
+start()
+setModel(IModel : model)
View
-model
-controller
1
-frame
1
View
+repaint()
+View()
+openFrame()
+closeFrame()
+getObserver(
+setController
GameFrame
-width : int
-height : int
+gameFrame()
+keypressed()
+keyTypes()
+keyreleased()
+repaint()
+getGamePanel() : GamePar
+getWidth() : int
+getHeight() : int
+getController() : Controller
+setController()
1
-panel
1
<<Interface>
Observer
+update()
GamePanel
+gamePanel()
+paintComponent
+update()
import
Java.keyEVENT
Java.Swing.JFrame
Java.awt.Graphic
Java.Swing.JFrame
Powered ByVisual Paradigm Community Edition

3.1.3 Controller

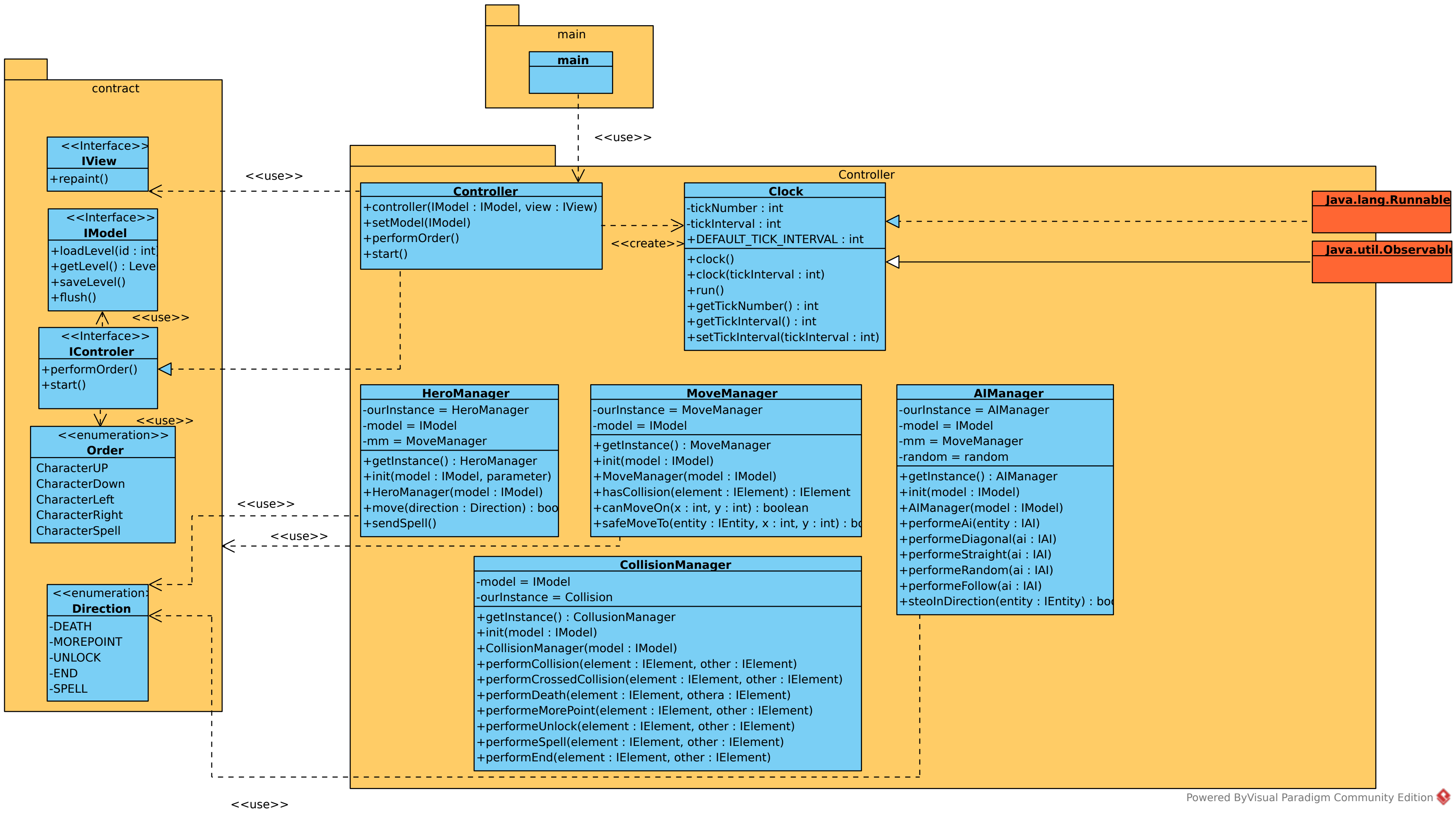
This part of the program is in charge of the event management. It has to synchronize every action in the software, in order update the view or the model correctly.

It receives events from the view unit and treat the information in order to trigger the appropriate reaction from the model part.

In our program the controller has to:

- Start the game
  - Controller class
- Synchronize the software's actions
  - Clock class
- Manage the behavior of every element that will appear on screen
  - HeroManager class
  - MoveManager class
  - CollisionManager class
  - AIManager class
- Inform the model of every order to perform and changes to apply
  - Controller class

to see a graphic view of the Controller unit please report to the next page.


main
main
<<use>>
contract
<<Interface>>
IView
+repaint()
<<use>>
<<Interface>>
IModel
+loadLevel(id : int
+getLevel() : Leve
+saveLevel()
+flush()
<<use>>
<<Interface>>
IControler
+performOrder()
+start()
<<use>>
<<enumeration>>
Order
CharacterUP
CharacterDown
CharacterLeft
CharacterRight
CharacterSpell
<<enumeration:
Direction
-DEATH
-MOREPOINT
-UNLOCK
-END
-SPELL
Controller
Controller
+controller(IModel : IModel, view : IView)
+setModel(IModel)
+performOrder()
+start()
<<create>>
Clock
-tickNumber : int
-tickInterval : int
+DEFAULT_TICK_INTERVAL : int
+clock()
+clock(tickInterval : int)
+run()
+getTickNumber() : int
+getTickInterval() : int
+setTickInterval(tickInterval : int)
Java.lang.Runnable
Java.util.Observable
HeroManager
-ourInstance = HeroManager
-model = IModel
-mm = MoveManager
+getInstance() : HeroManager
+init(model : IModel, parameter)
+HeroManager(model : IModel)
+move(direction : Direction) : boo
+sendSpell()
MoveManager
-ourInstance = MoveManager
-model = IModel
+getInstance() : MoveManager
+init(model : IModel)
+MoveManager(model : IModel)
+hasCollision(element : IElement) : IElement
+canMoveOn(x : int, y : int) : boolean
+safeMoveTo(entity : IEntity, x : int, y : int) : bo
AIManager
-ourInstance = AIManager
-model = IModel
-mm = MoveManager
-random = random
+getInstance() : AIManager
+init(model : IModel)
+AIManager(model : IModel)
+performeAi(entity : IAI)
+performeDiagonal(ai : IAI)
+performeStraight(ai : IAI)
+performeRandom(ai : IAI)
+performeFollow(ai : IAI)
+steoInDirection(entity : IEntity) : boo
<<use>>
<<use>>
CollisionManager
-model = IModel
-ourInstance = Collision
+getInstance() : CollusionManager
+init(model : IModel)
+CollisionManager(model : IModel)
+performCollision(element : IElement, other : IElement)
+performCrossedCollision(element : IElement, other : IElement)
+performDeath(element : IElement, othera : IElement)
+performeMorePoint(element : IElement, other : IElement)
+performeUnlock(element : IElement, other : IElement)
+performeSpell(element : IElement, other : IElement)
+performEnd(element : IElement, other : IElement)
<<use>>
Powered ByVisual Paradigm Community Edition

3.1.4 Contract

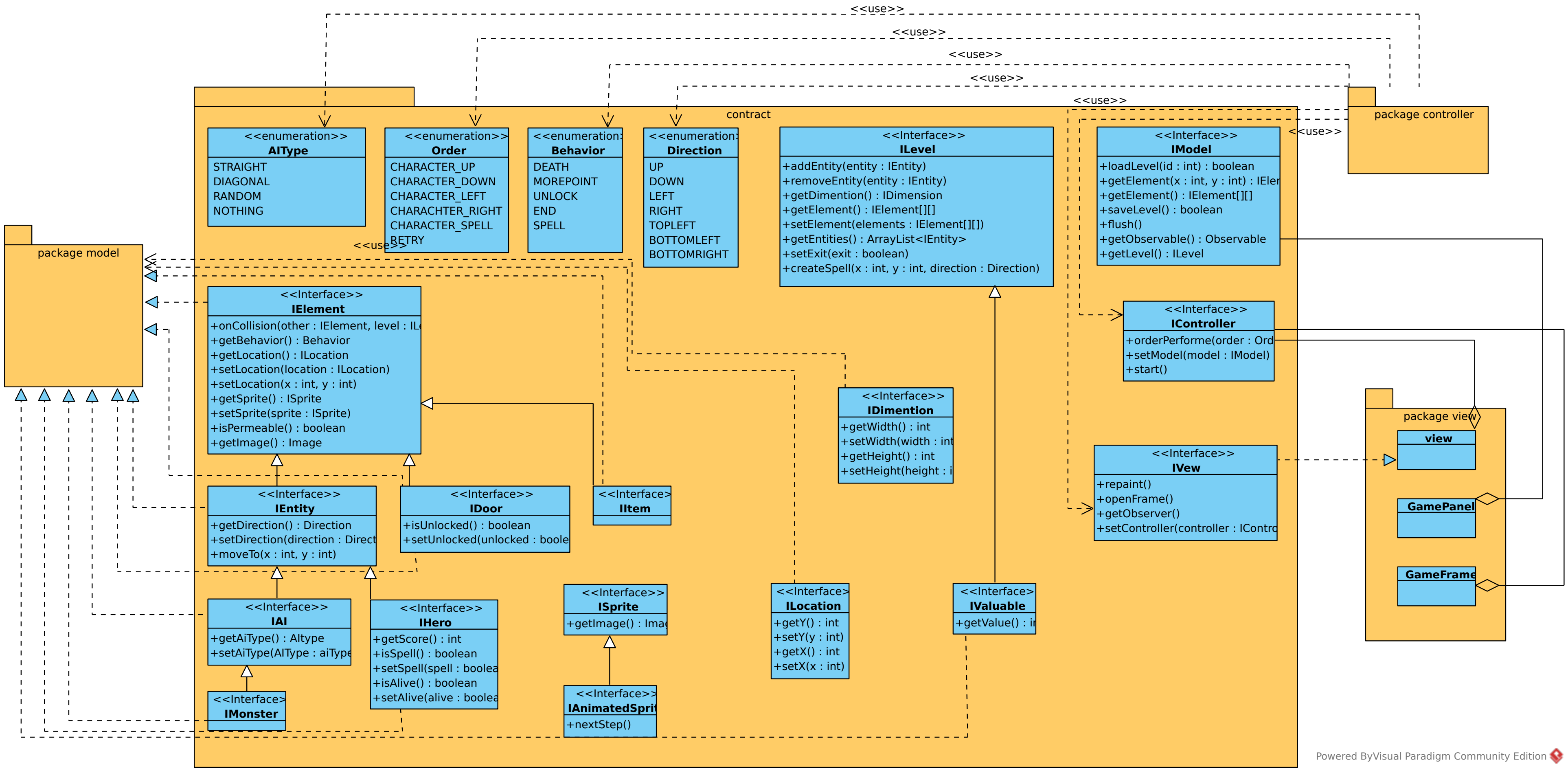
A classic MVC pattern has a default, it creates a large number of couple between the three parts, that may cause program dysfunction and it decrease the code's reusability.

In order to face those problems, wet set a fourth unit in our program, the contract.

In this last part we create many interfaces within which contain operation from every class that has to be use in another part of the program than its native unit.

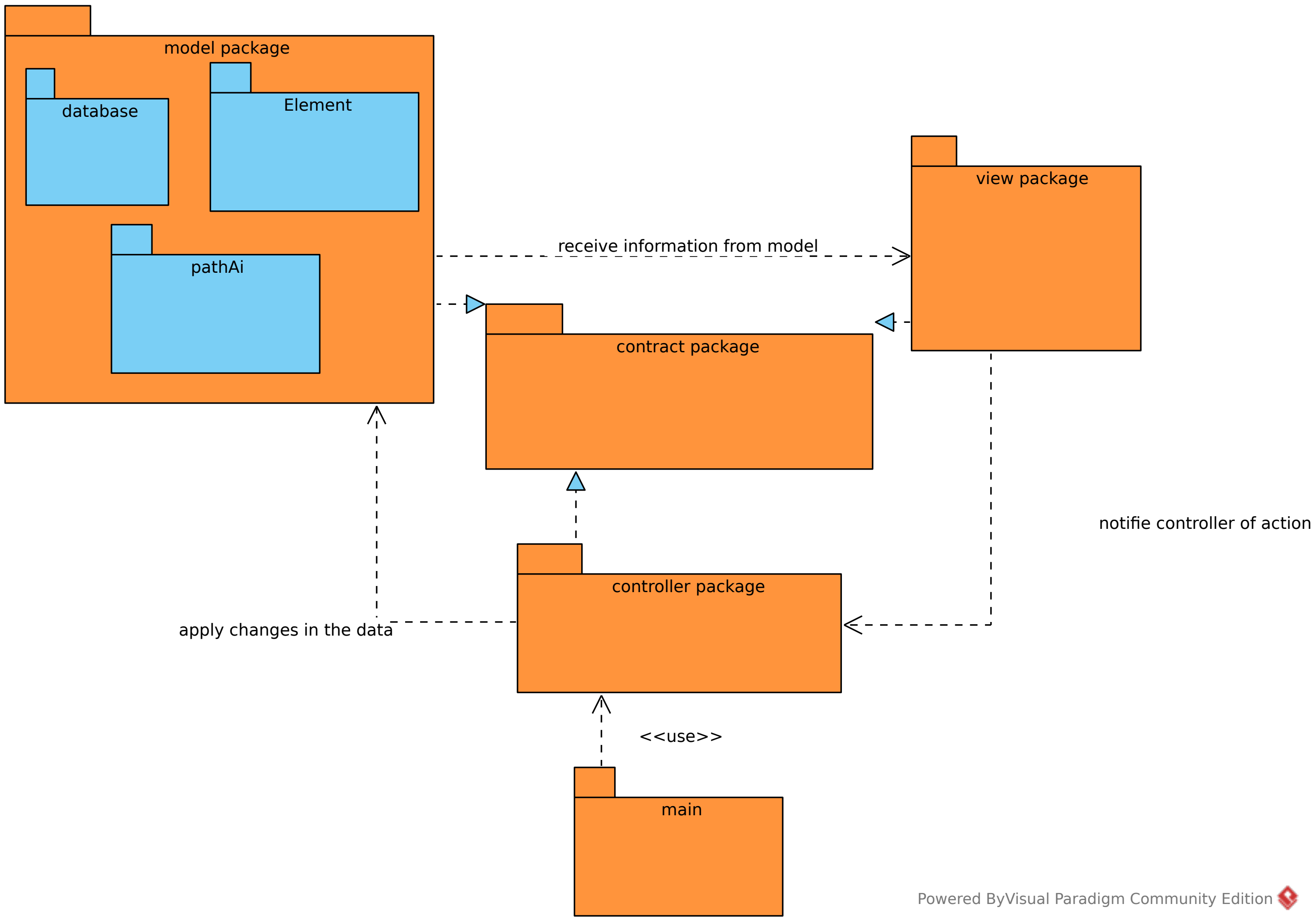
For example, the view needs information from the model, so it will use its interface, IModel.

to see a graphic view of the Contract unit please report to the next page.



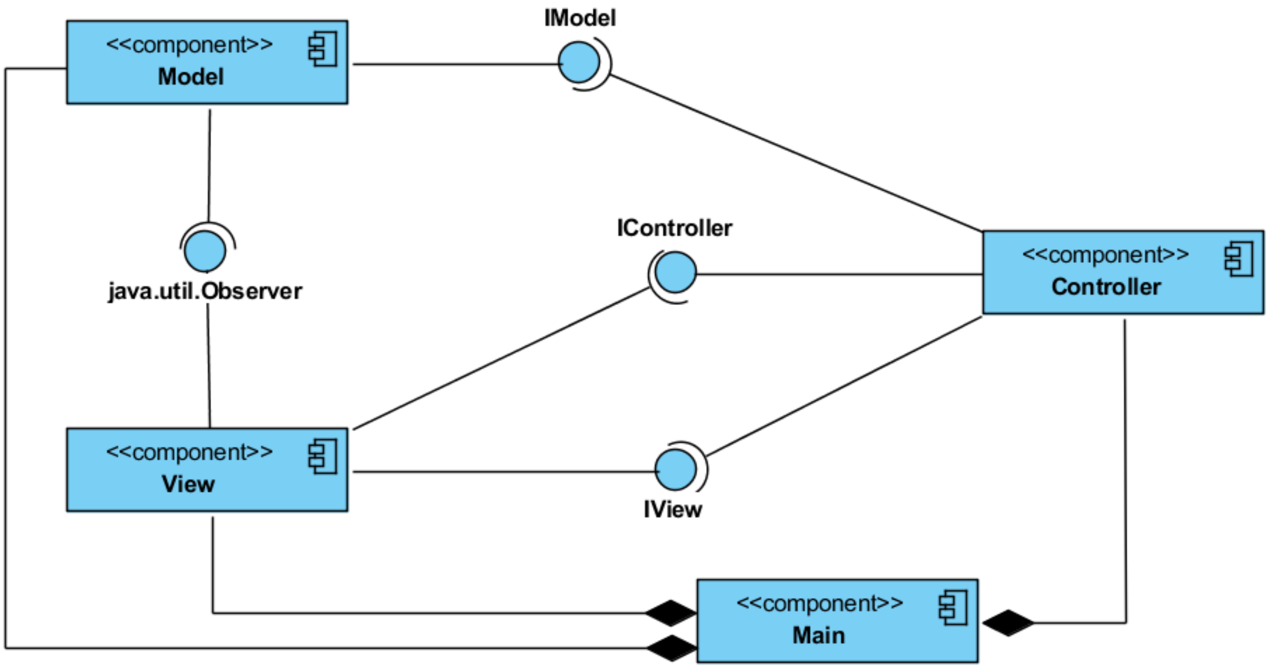
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
contract
package controller
package model
package view
<<enumeration>>
AIType
STRAIGHT
DIAGONAL
RANDOM
NOTHING
<<enumeration>>
Order
CHARACTER_UP
CHARACTER_DOWN
CHARACTER_LEFT
CHARACHTER_RIGHT
CHARACTER_SPELL
RETRY
<<enumeration>
Behavior
DEATH
MOREPOINT
UNLOCK
END
SPELL
<<enumeration>
Direction
UP
DOWN
LEFT
RIGHT
TOPLEFT
BOTTOMLEFT
BOTTOMRIGHT
<<Interface>>
ILevel
+addEntity(entity : IEntity)
+removeEntity(entity : IEntity)
+getDimention() : IDimension
+getElement() : IElement[][]
+setElement(elements : IElement[][])
+getEntities() : ArrayList<IEntity>
+setExit(exit : boolean)
+createSpell(x : int, y : int, direction : Direction)
<<Interface>>
IModel
+loadLevel(id : int) : boolean
+getElement(x : int, y : int) : IEler
+getElement() : IElement[][]
+saveLevel() : boolean
+flush()
+getObservable() : Observable
+getLevel() : ILevel
<<Interface>>
IController
+orderPerforme(order : Ord
+setModel(model : IModel)
+start()
<<Interface>>
IVew
+repaint()
+openFrame()
+getObserver()
+setController(controller : IContro
view
GamePanel
GameFrame
<<Interface>>
IElement
+onCollision(other : IElement, level : IL
+getBehavior() : Behavior
+getLocation() : ILocation
+setLocation(location : ILocation)
+setLocation(x : int, y : int)
+getSprite() : ISprite
+setSprite(sprite : ISprite)
+isPermeable() : boolean
+getImage() : Image
<<Interface>>
IDimention
+getWidth() : int
+setWidth(width : int
+getHeight() : int
+setHeight(height : i
<<Interface>>
IEntity
+getDirection() : Direction
+setDirection(direction : Direct
+moveTo(x : int, y : int)
<<Interface>>
IDoor
+isUnlocked() : boolean
+setUnlocked(unlocked : boole
<<Interface>
IItem
<<Interface>>
IAI
+getAiType() : AItype
+setAiType(AIType : aiType
<<Interface>
IMonster
<<Interface>>
IHero
+getScore() : int
+isSpell() : boolean
+setSpell(spell : boolea
+isAlive() : boolean
+setAlive(alive : boolea
<<Interface>>
ISprite
+getImage() : Imag
<<Interface>>
IAnimatedSprit
+nextStep()
<<Interface>
ILocation
+getY() : int
+setY(y : int)
+getX() : int
+setX(x : int)
<<Interface>
IValuable
+getValue() : in
Powered ByVisual Paradigm Community Edition

3.1.5 Packages


model package
database
Element
pathAi
view package
receive information from model
contract package
notifie controller of action
controller package
apply changes in the data
<<use>>
main
Powered ByVisual Paradigm Community Edition

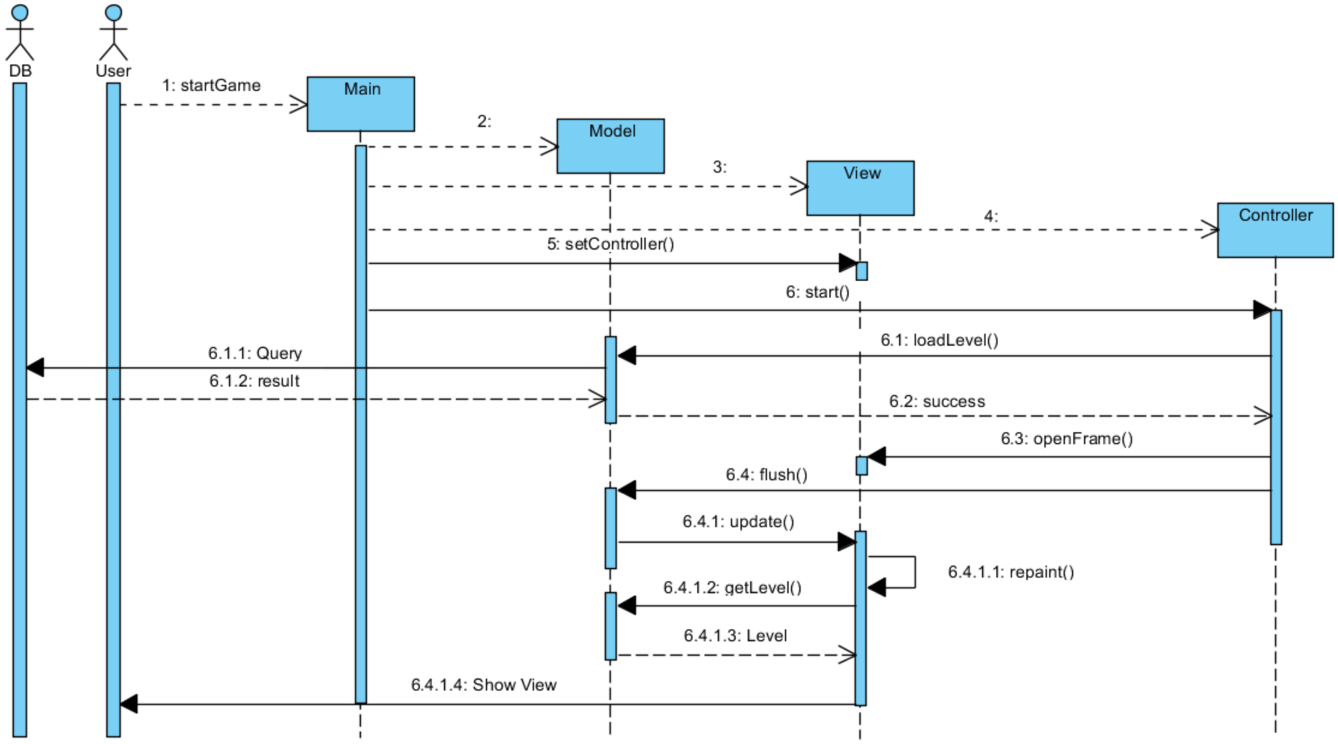
3.2 Components

Finally the program's components are connecting this way



3.3 Sequence in game

The program is obeing to the following diagram.

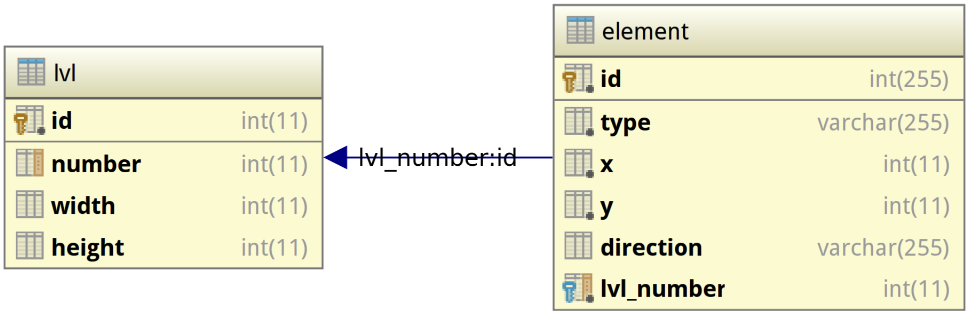


First the user starts the game, the main will activate the model, the view then the controller. The controller will ask the model to load the first level, the level is charged from the data base and return to the model. Next the controller will make the view open a frame that will the interface with the user. At every player's action the view will update by charging the model's data.

3.4 Database

3.4.1 Map

To create the maps first we made a text file were each letter was corresponding to an element of the game. Then with a filling script we fill the database with all the elements and their location (x,y) on the different levels of the game.



lvl	
id	int(11)
number	int(11)
width	int(11)
height	int(11)

element	
id	int(255)
type	varchar(255)
x	int(11)
y	int(11)
direction	varchar(255)
lvl_number	int(11)

3.5 Stored procedures

The database is composed of 3 stored procedure to create de different levels.

get_elements_by_level return all the elements from a specific level

```
DELIMITER |
CREATE PROCEDURE get_elements_by_level (IN lvlID int)
        BEGIN
                SELECT element.id, element.type, element.x, element.y, element.direction, element.lvl_number FROM element WHERE lvl_number = lvlID
        END |
DELIMITER ;
```

get_levels return all the levels

```
DELIMITER |
CREATE PROCEDURE get_levels
        BEGIN
                SELECT *
                FROM lvl
                ORDER BY lvl.number
        END |
DELIMITER ;
```

get_level_by_id return a specific level

```
DELIMITER |
CREATE PROCEDURE get_level_by_id (IN id int)
        BEGIN
                SELECT * FROM lvl WHERE lvl.id = id LIMIT 1
        END |
DELIMITER ;
```

Chapter 4

Reports

4.1 Git

During this project we made over *180* commits on our repository on GitHub at `goo.gl/EyqrgK`

4.1.1 Contributors participation

This is the contribution of every member of the team based on the number of commits they made.

Clément	28,5 %	**Marie**	15 %	**Baptiste**	68,8 %	**Maxime**	3,7 %

Chapter 5

Personnal assessment

5.1 Clément Chabrier

This JAVA project was very interesting yet complicated, I thought that my lacks in this language would be penalizing for the group.

However, with our teamwork we managed to compensate for everyone's weaknesses and reach our goals.

Through my mission of building the view part I improved my skills on several point about the JAVA language, specifically on software architecture and visual shows on JAVA console.

Also I developed a real complicity with my colleague, Baptiste Saclier, which led us to an optimal development of the project.

5.2 Maxime Zupka

This project was very interesting because it was a very comprehensive one. We had to use Java which is a new language but also other things that we learn during the year such as database. And I really like the idea of remaking an oldschool game. I think we did a great a job and the project is a success.

5.3 Baptiste Saclier

This project was an opportunity to me to discover the happy things of the team managment and the Java project build. I think all my team was involded in the creation of this game and they wants all to participate.

To conclude, it was a great project with many challenges.