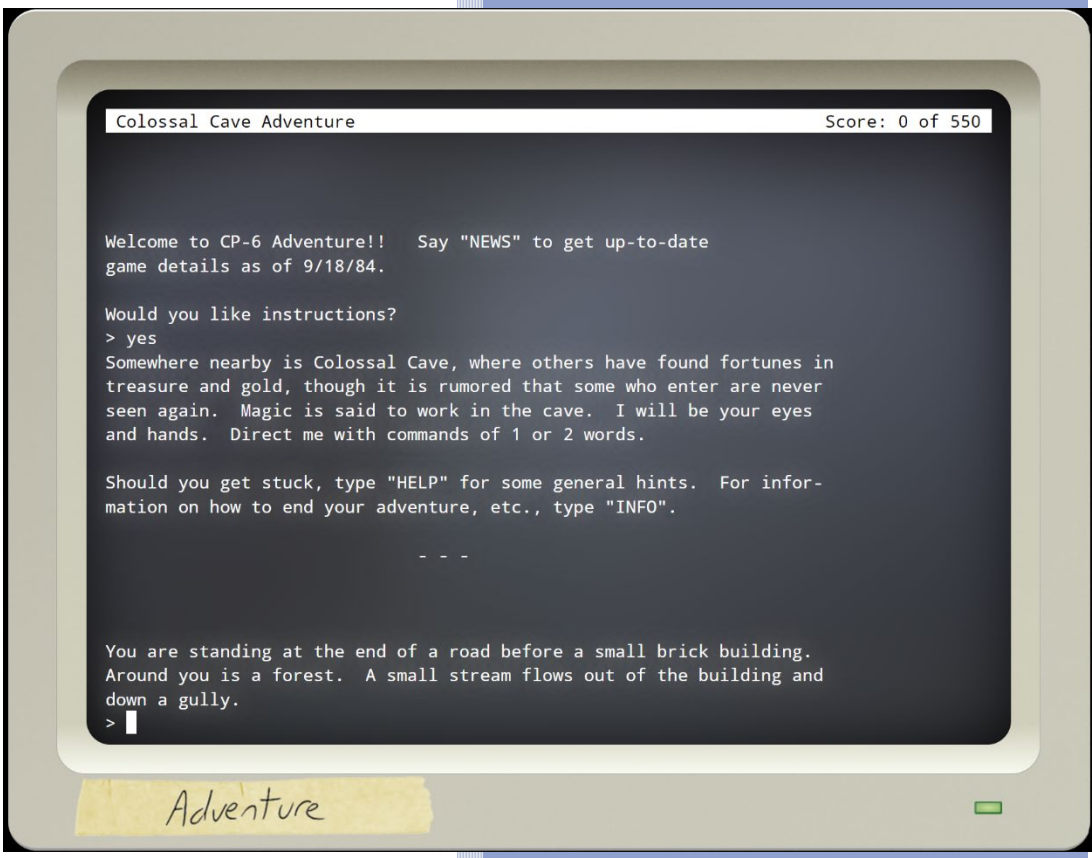
2022

Rapport de Projet



MEDEWOU Selly

Université de Poitiers

Table des matières

User Documentation

Presentation of the game

This is a text-based adventure game in the style of Colossal Cave Adventure. You will progress through a world that you will imagine through the texts that will be displayed on the screen and by interacting using commands given in the game.

Synopsis

The game takes place in the University of Poitiers. You land there without knowing why and the goal is to escape, but be careful, there are several obstacles that prevent you from doing so, such as monsters and puzzles.

Lists of commands

There is a list of commands that you can also find in the game by typing the help command.

Take : argument : adds (if possible) the argument to the hero's items.

Example : Take coat

Load :

Allows you to load a previous save...

Example : Load

Help : indicates the set of availaible commands.

Example : Help.

You can actually add another argument to display informations of a particular command Example : Help Go

Go : the command GO is followed by the name of the neighbor location the player wants to go.

Example : Go Bathroom

Holding :

Shows all items in your hands.

Example : Holding

Disarm :

Allows you to put your equiped weapon into your bag

Example : Disarm

Check :

Allows you to search furniture.

Example : Check desk

Quit : quit the Game.

Example : Quit

Check_bag :

Shows all items inside your bag, but only items not hints.

Example : Check_bag

Attack :

Allows you to attack all enemies in your current location.

Example : Attack

Equip : uses the object you have typed.

Example : Equip gun, will equip your player with the gun

Throw :

Allow you to throw away an item from your bag to your current location.

Example : Throw gun

Monster :

Shows all monsters including their stats present in your current room

Example : Monster

Clues :

Allows you to see all clues that you have gathered...

Example : Clues

Save :

Allows you to save the game ...

Example : Save

Look : displays a description of the the current location and all items

Pack :

Allows you to pack the item in your hand into your bag.

Example : Pack

My_stats :

Allows you to see your current stats like your name, life points, damage, and the volume of your bag.

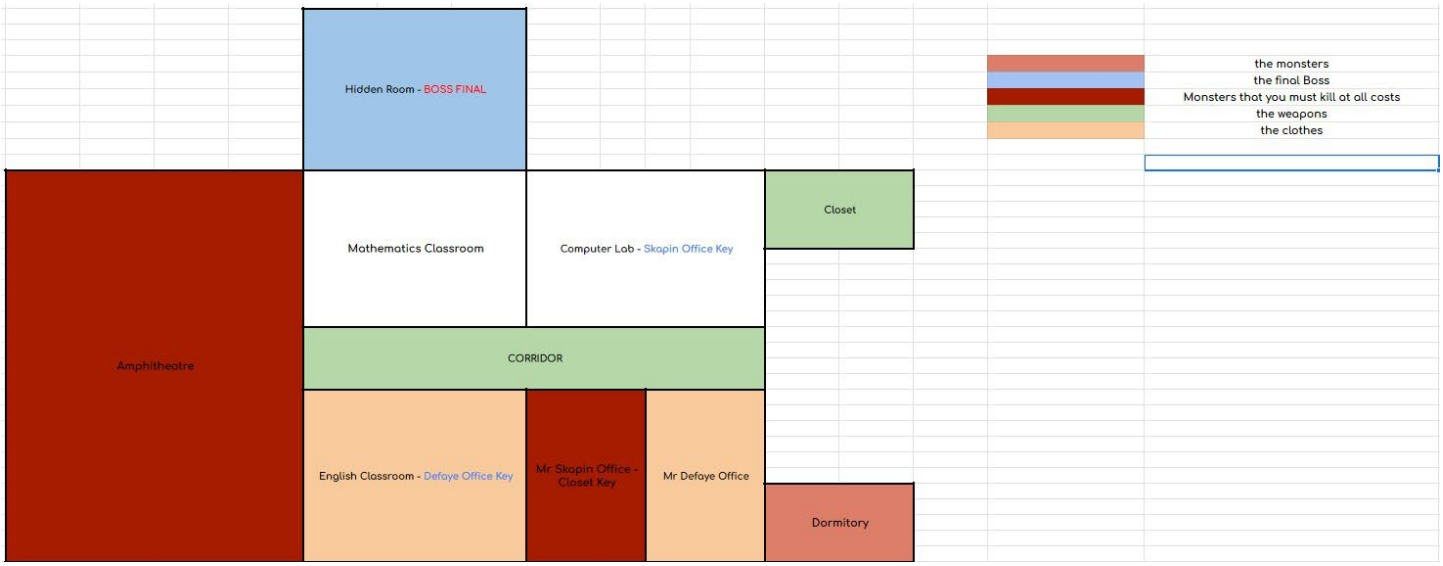
Example : My_stats

See_rooms :

Shows all adjacent rooms

Example : See_rooms

The Map of the University



**the monsters* : "It is the color of the location that you could find some monster to fight"

*The Final Boss** : "Like it's said, that is the place where you could find some boss"

**Monsters that you must kill at all costs* : "This means that if you enter these locations, you will only be able to leave if you manage to kill all the monsters present there."

**The weapons* : "It is the color of the location that you could find some weapon to take "

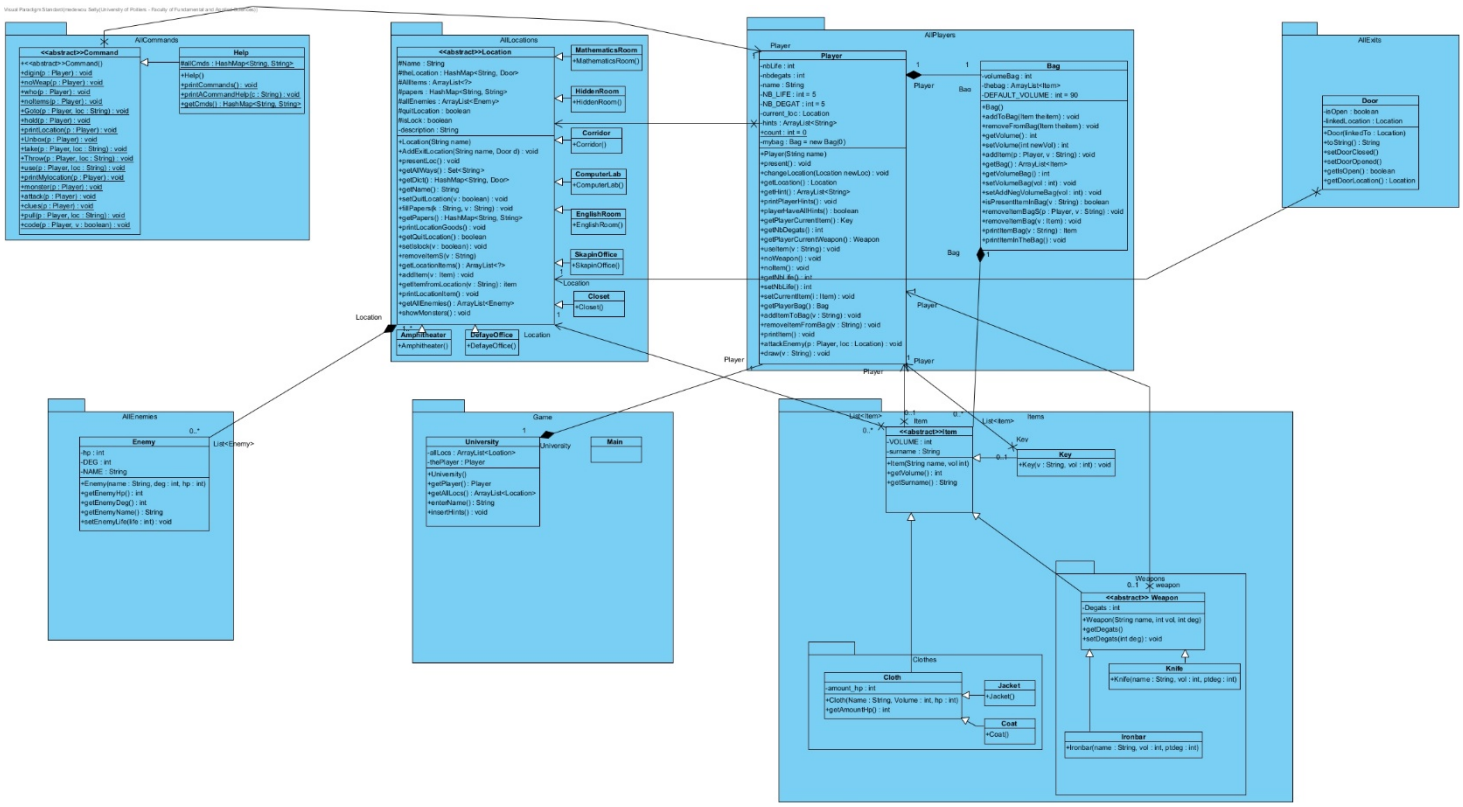
**The clothes* : " It is the color of the location that you could find some clothes to take "

Use of the game

To install and launch the game, simply run the command : "sh LaunchGame.sh" in the Linux shell. However, the version with tests does not compile with javac. We have an error message indicating that the package org.junit.jupiter.api does not exist. This version compiles correctly in Eclipse and the tests work there. As a result of this problem, we created two folders : one with tests and the other without tests with an executable.

Programmer documentation

Diagram UML



How I handle the link between the Location's class with the Exit's class

Each location contains a set of exits, allowing the player to go from the current location to a nearby one. Of course, different kinds of exits may exist (some can be always opened, other may need a key).

So, each exit knows its neighbor location. Otherwise said, the player can try to cross

any exit, but this will not be done the same way if the exit is a simple door or a door which needs a key.

Note :

" The game handles one-way exits (i.e. a location

A may contain an exit for going into location B, but nothing guarantees that once the

player is in location B there will be an exit for going back to location A).“

From a given location, it will therefore be possible to go to a neighboring location provided you are able to cross the corresponding exit. All the exits of a location will be stored as a HashMap<String,Door> allowing to associate each neighboring location with its name. To go to a neighboring location B, you will therefore have to cross the asso-ciated exit using the command GO name_of_location_B (see List of Commands section).

```
public Location(String name)
{
    this.Name = name;
    this.theLocation.put(this.Name, new Door( linkedTo: this));
    this.quitLocation = true;
    this.isLock = false;
}
```

So the location class, has an attribut theLocation of type hashMap<String, Door> where the first time we create a location, we have a door to go out of it.

And then if we create another location B, to relate it to another location A, we just need to add to the « theLocation>attribut of location B, the name of location A and new Door(the location A), and repeat the same process for the location B in Location A…

For more information, you can take a look of the University class and try to understand what Have been Done…

Thanks you…