Save the Most Precious Thing

By Yajing Wang & Mikhail Korotych

About the Game

This is a text adventure game set in the Louvre Museum in Paris, France. You must enter the correct answer to advance the action or location. The correct answer is no more than two words and is often a noun, a verb, or a combination of a verb and a noun. At the same time, pay attention to the items that appear in the scene, since the time of session is restricted. When you pick up an item, this item will be removed from the current location. When you throw an item away, this item will be placed on the current location. When you meet a non player character (NPC), you can talk or ask with them to get more information.

Introduction

Today is Monday, the closing day of the Louvre. The sun sets in the west, and the golden sunset shines on the huge glass pyramid in front of the Louvre. Although you have seen this beautiful scene many times, the sunset still makes you deeply addicted every time. While you were immersed in the scenery, you received a message on your phone…

The Set of Input Commands

The set of input commands is not strict, so there are a few options to designate what you need to do during the game. The response for the unappropriated command in some situation is like “It seems like this is not the correct direction…” or “This is not the correct command.” So, here is the list of the input commands currently supported by the game. Not all of them are available at every location. All the text in square brackets is optional but if you are typing it, pay attention to the vertical line which means “or”.

- `[go|move] north|south|east|west|back`;
- `look [around]|observe|explore`;
- `take|pick` an item;
- `drop|throw` an item;
- `read` a readable item;
- `[check]` inventory;
- `help`.

Architectural decisions

You may see them in the UML diagram below. There are highlights of them.

1. An abstract class `Character` to represent its children: the main user, NPCs, so-called “friends”. Some of them have their own inventory (the user, an NPC). The user has also a

stack of visited locations while each location has 4 neighbors (one location per a side of the world) like a graph node.
2. Such game items as the notebook and the museum map extend `AbstractItem` and implement `IApplicable` to be *applicable*, e.g. printing messages while user interacts with them. Non-applicable items which do not implements `IApplicable` interface (for instance, the newspaper) cannot print such messages.
3. There are a lot of locations in this game. All of them extends the base class `Location` in terms of printing messages when the user enters to a new location. All location has also a method to print special messages within the map and their own inventory where different items can be stored. Of course, the location cannot modify its inventory by itself, but the user can by taking an item from the location or dropping an item on it.
4. Parsing and processing of the user input is provided by the `MainLoop` class while the `Command` enumeration contains all commands the game can handle.
5. The global and static class `TimeCounter` is needed to set time restrictions (we are trying to save the **most** precious thing, right?). These restrictions include the maximum of the in-game time in minutes, a number of minutes per a step and the number of a step which corresponds to the command of taking a step.

Workflow

During the period of working on our game, which we decided to name “Save the Most Precious Thing”, we had an intense exchange of thoughts and ideas about the game plot, rather techniques and possible characters.

At the beginning two versions of the UML was drawn (Mikhail’s and Yajing’s) and the one by Yajing was chosen as those to base further development on.

Then we split our code space in two branches to follow Git Flow rules for effectiveness and start to code entities but, unfortunately, because of hard going processes of solving merge conflicts we decided to develop our game at the `develop` branch one by one, commits by commits.

Mikhail’s contribution

I developed the inventory and operations with it, a logic of items and in-game time counter.

While the inventory is just a wrapper over the Java array, the behavior of items is more complex. As it was mentioned before they can be applicable and non-applicable with the help of `AbstractItem` and `IApllicable` but at the beginning there was also `INonApplicable` interface.

I would like to highlight that I had to make the `getItem` method in the `Inventory` class so-called covariant to take both applicable and non-applicable items.

But we thought this pattern is too complex since we may use just `AbstractItem` and `IApllicable` to follow applicability.

The counter of the in-game time counts, to be honest, not the time itself but the user’s steps and converting it to the in-game minutes with the help of predeclared constants. That is why the game time follows only when the user takes a step.

I like our project! I have reinvented the team development. The smaller the development team (especially if it’s a non-profit or pet project), the more complex the development.

Communication in a non-native language can bring even more difficulties and misunderstandings, but we learn to overcome such moments.

Yajing's contribution

I mainly conceived the main plot of the game, and most of the instructions in the game are original to me. This game was born out of my experience of traveling to the Louvre and a TV debate I once heard.

In terms of code writing, I mainly wrote two classes, `Game` and `Mainloop`, as well as most codes for `Characters` and `Locations`. Although I wrote the main structure of the game, during the coding process, Mikhail and I were discussed and resolved together when we encountered problems.

I really like the cooperation process between the two of us. It is very pleasant and smooth because we both try hard to understand and listen to each other. I am very grateful to be in a group with Mikhail.

Manual

This is a quick answer to complete the game, but it's not the only correct answer:

"about" → "ready" → "read message" → "any key" → "museum" → "denon" → "look" → "north" → "pick map" → "read map" → "go north" → "look" → "pick notebook" → "read notebook" → "look" → "go east" → "look" → "go east" → "read map" → "go Madonna" → "ask" → "no" → "read key" → "open door" → "save kitten/Monalisa".

During the game, especially after entering the Denon Hall and before finding the Monalisa, you can always type `help`/`inventory`/`quit` to find more information and `exit` from the game.

Map for the Game

Please, look for the "Map.jpg" file in the repository.

The UML

Please, look for the "UML.svg" file in the repository.