Polytech Sorbonne

Project report

Yellow — C++

Authors :
Sylviane Ly
Thizirie Ould Amer

Supervisors :
Cécile Braunstein
Jean-Baptiste Brejon

January 21, 2019

Contents

1 Introduction

Our C++ project is about art. It challenges you at your knowledge of all types of works. When you start the game, you have 5 points. Through questions about paintings, sculptures... you can loose them or in the better case increase your score. Let see how it works.

So you think you know art ? What about **Yellow Art** ?

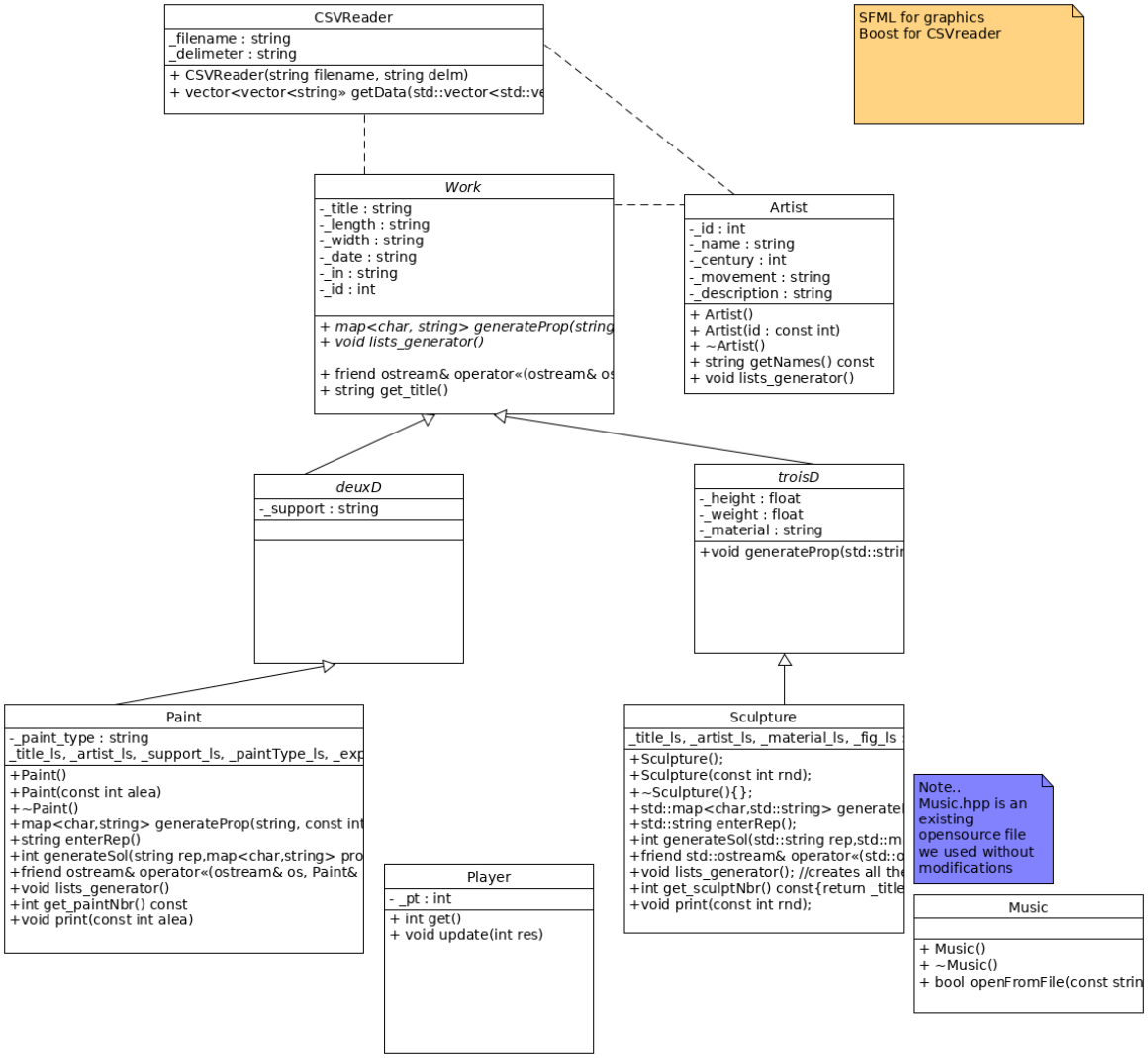
Want to give it a try? Go to the instructions: 5

2 Description

Like all good quizzes, there are questions. When we start the game, an artwork is randomly chosen form a little database and the computer challenges our knowledge about this object. For example we can have a question about the title of a painting. Next we can have questions about the paint type that was used, its author, the date... It can also change the work and ask for the title of a sculpture...etc.
This game is only available with a musical background!

So, first, turn on the volume! Then choose your favorite singer/band when the game asks you. The game begins right after your answer!

3 UML Diagram



4 Implementation

As you can see on the diagram, the main classes are : CSVReader, Work, Artist, Player and Music. We are proud to give some details about the CSVReader and `GenerateProp()` since they constitute the minimum and fundamental parts of our code.

4.1 CSVReader

csvreader.h is a class we implemented to make it easier for us to read different files (since we have three of them; containing : Paints, Artists and Sculptures). The default

delimiter is a comma (,) but could be changed, of course. This means that this class is going to be useful in the future even for different files extensions, with some simple and minor changes.

The way it is built makes files manipulation easier. With the variable *count*, we make sure that the first line of each file is not read (we do not want to save the titles of each column).

```
void CSVReader::getData(std::vector<std::vector<std::string>>& dataList){

        std::ifstream file(_filename);
        bool count = 0; // used to skip the first line of the csv

        std::string line = "";

        // Iteration through each line and splitting the content
        //using the specified (or default) delimeter
        while (getline(file, line))
        {

            if (count)
            {
                  std::vector<std::string> vec;
                  boost::algorithm::split(vec, line, boost::is_any_of(_delimeter));
                  dataList.push_back(vec);
                  }

                  if (count==0)
                          count = 1;
        }

        // Close the File
        file.close();
}
```

In the while loop, it reads each ligne of the csv and separates each column's elements by the delimiter (default : a comma). Then, every ligne (which is a "vector") is stored in a vector (here, called *dataList*). So dataList is of type *vector¡vector¡string¿¿*.

4.2 GenerateProp()

This function has two arguments : a string (the question) and an integer (randomly choosen in main).

Each question is clearly identified by a letter : T for title, A for artist, S for the paint type (or M if we are dealing with sculptures). Depending on the question, it fills the correct list (list of titles...) in a vector. Then we select 5 propositions among the vector to print them on the screen. We use a map to do it because one proposition is a letter and is associated to a string in the correct list.

An important thing must no be forgotten : the right answer has to figure among the propositions. In order to do that, we have decided to replace a proposition by the true

answer that is to say by the attribute of the object. This is what the code here is doing.

```
//Making sure the correct answer is always displayed in the choices
char i = char(rand()%4+65);      // i = A, B, C, or D

//Painting title questions
if (code == 'T'){
        std::map<char, std::string>::iterator it = prop.find(i);
        if (it != prop.end())
                (*it).second = _title_ls[rnd];
        }

//support type questions/Paint type
if ((code == 'S') && (_paintType_ls[rnd]!="")){
        prop[i] = _paintType_ls[rnd];
}

/*Artist questions. We have to change here what's displayed,
from just an id of the artist to her or his name*/
if (code == 'A')
{
        prop['A']=names[std::stoi(prop['A'])];
        prop['B']=names[std::stoi(prop['B'])];
        prop['C']=names[std::stoi(prop['C'])];
        prop['D']=names[std::stoi(prop['D'])];
        prop[i]=names[std::stoi(_artist_ls[rnd])];
}
```

The integer `i` takes a proposition at random thanks to the letter that we have affected to each proposition. It overwrites the object's attribute instead of the proposition. We have three ways of implementation but they all do the same thing.
For the title and support questions, a number (`rnd`) locates the position of the painting in the corresponding list. However in the last case, the position in the list is determined by the number given by `stoi()`. An artist is assigned to an id, that is why we can pass from one to another with the list `names`.

4.3 Dealing with constraints

Here are the constraints and how we dealt with each one of them:

- 8 classes (at least) → csvreader, Artist, Work, deuxD, Paint, troisD, Sculpture, Player, Music.
- 3 hierarchy levels → Paint inherits from deuxD which inherits from Work. Sculpture inherits from troisD which also inherits from Work.
- 2 virtual functions → `GenerateProp()` and `lists_generator()`. They are respectively used by Paint, Sculpture and Artist.
- 2 overloaded operators → There was no usefull cases to overload +,—— ...etc. The operator "<<" is overloaded twice, for paintings and sculptures.
- 2 containers (from STL) → We use a lot `vector` and also `map`.
- A class diagram → See 3.
- No errors with Valgrind → Unfortunately we have some problems with the memory, probably due to the sfml library. Indeed, when we disable the music and the window to display paintings, there are 0 errors.

Moreover our functions had to stay short enough. Ours actually exceed 30 lines, but mostly to have a more readble code (empty lines, many comments...).

5 Installation

Installation of some librairies :

- boost : `sudo apt-get install libboost-all-dev`
- sfml : `sudo apt-get install libsfml-dev` (for the music and the pictures)

Then you can compile with the Makefile and execute it normally writing

- `make`
- `./main`

6 Conclusion

This is already the end. After a few words about our Yellow Project. We have shown you the key elements starting with the class diagram and some important functions. We have almost overcome all the constraints. To improve this game we can do all of it in a graphic interface. The databases can continue to be supplied with paintings, sculptures but also with other art forms like short movies, songs...

If we have the time for, we would try to put on some additionnal options, like showing an artist with his description, at the end of the game. We can also try to make a real 2-dimentional game, with some players walking through an art gallery (we thought about choosing one of the four Beatles to play with, then exploring a yellow submarine full of yellow artwork).

We hope you enjoyed playing the game and would love to have your feedback and maybe ideas to improve it!