AG0701A Programming in C++ - coursework

Emilian Cebuc

C++ REPORT

Assignment approach

The approach to this program was based on the existence of the two entities of the game, the players and their actions aimed to win the game as fast as possible. From the very beginning the use of Object-Oriented Programming showed its utility, as defining stats for the players, "actions" (the various throw types) needed to be done once, but the core of the experience was characterized by the sense of essentially having two totally different players, actually thinking and acting for themselves individually, and be able to make decisions and choose the best next move to counter the opponent's attempts or even simply deal with their own score situation.

Once the players were defined, the next most important and difficult thing to start conceiving was the whole AI system. It started very simple (trying to get the score to 0 in the fastest way possible), subsequently adding more complex and strategic conditions for the score in order for the player to act accordingly and always keeping efficiency in mind instead of simple blind attempts to the maximum number.
Players concentrate first of all on the doubles leaving doubles, trying to end with them, as a failure (hitting the single) still leaves a double at the next throw. They try to aim for the differences from these "hot" positions in order to get as close as possible to them. The AI system also calculates in advance combinations of doubles and trebles for cases in which the player has the possibility to end fast in 2 or 3 throws, as they both try (especially when both have high accuracies) not to risk giving the other the chance for another turn (thus 3 attempts) end perhaps end before them.

Once the AI system was done, other, "optional" features were added, such as input mistakes handling (preventing the game to enter infinite loops when weird characters are inserted), the usage of the Mersenne Twister pseudo-randomizer for more realistic "more random" random numbers, 3 different possibilities of setting players' accuracies, both for the simulation and the interactive game (essentially allowing the user to choose a difficulty in the latter game type) as well as choice of a standard 501 game or an extended 701 game. Finally, an important part was played by the whole graphical overhaul, using snippets and functions from the IaMothe code for repositioning the outputs in the game and actually displaying static players' stats and throw updates for both players, as well as an indicative graphical "dartboard".

Object-Oriented Programming vs Procedural Programming

In comparison with the first semester this new style of programming and organizing data clearly allows for a neater and more efficient structuring of a program.
Thanks to the use of setter and getters it is possible to create a uniformed interface in order to interact with the objects' data members and also impose a basic layer of permissions to the methods and functions, this way increasing the overall security. It also more easily enables the programmer to organize the files of the project, allowing for example to split every class into its own header and code files, therefore enabling the coder to easily share these structures among every source file that might require them.

As far as cons are concerned, it can require quite a bit of code just to be able to perform the basic functions (I.E. define the classes, getters, setters when in procedural it would simply be necessary to create and assign a bunch of variables).