AG0701A Programming in C++ - coursework

Emilian Cebuc

DARTS GAME PSEUDOCODE

Create Player class

Create header file to contain the class information
Declare player private data fields: name, bull and single accuracies, score
Declare player public fields: games, sets, matches won, number of turns
Declare the setters and getters for the above private fields, and define them all in the class .cpp file
Declare and define the function for setting all accuracies based on a master number passed to it
Declare and define the bull() hit function

- Calculate a random number from 0 to 100 (mt19937 algorithm with chrono for efficient randomization)
- If the random number is smaller or equal to the bull hit percentage, return 50
- Otherwise if the random number is between the bull hit percentage and a certain other value return 25
- Otherwise, randomly hit any of the 20 singles of the board and return that value hit

Declare the single() hit function

- Calculate a random number from 0 to 100 (using same mt19937 randomization)
- If the chosen target number is the outer bull
  - Divide the accuracy difference between bull hit % and 100 in 3 parts
  - Check if the random number is between the bull hit % and the first part; if yes, return 25
  - Otherwise if the % is between the first and the second interval, then return a bull (50)
  - Otherwise hit one of the 20 singles chosen at random
- Otherwise
  - Divide the accuracy difference between single hit % and 100 in 4 parts
  - If the single hit % is smaller or equal to the random number, then return that number,
  - Otherwise if the random number is between the bull hit % and the first interval, return the single at the right of the target number,
  - Otherwise if the random number is between the first and the second interval, return the single at the left of the target number
  - Otherwise if the random number is between third and fourth interval, return the double of the target
  - Otherwise hit the treble of the target

Declare the double function

- Calculate random number from 0 to 100 (using same mt19937 randomization)
- Divide the accuracy difference between single hit % and 100 in 3 parts
- If the random number is smaller or equal to the single hit % return the target number, doubled
- Otherwise if the random number is between the single hit % and the first interval, return the target
- Otherwise if the random number is between the first and second interval, return number at the left, doubled
- Otherwise return number at right, doubled

Declare the treble function

- Calculate random number from 0 to 100 (using same mt19937 randomization)

Divide the accuracy difference between single hit % and 100 in 4 parts
If the random number is smaller or equal to the single % then return target number, tripled
Otherwise if the random number is between the single hit % and sum of first 2 intervals (bigger chance because 2 options for hitting single) return the target
Otherwise if the random number is between second and third intervals, return number at the left, tripled
Otherwise return number at right, tripled

Initialize the player score to 501 (default score), all other data members to 0, inside the Player class constructor

Main() program

Declare and initialize to null variables for the names, accuracies, choices, the board target number
Declare a list of numbers for the dartboard and fill it with the various single values, in correct clockwise order
Declare another list for the double hits leaving numbers that are still doubles, fill it in descending order
Declare a list of numbers to contain all the possible winning sets combinations for the players
Declare a string list containing the written combinations to be output at end of simulation
Declare a list Player, and instantiate two objects (the players)

Allow the user to choose between a simulation of the computer fighting itself, and an interactive game
For both the simulation and the interactive game, let the user input names for the two players with a for loop
Prompt the user to select one of 3 methods to set accuracies for the players: manually inserting values, inserting a master value that decides each individual accuracy, or select the master value from a fast-incrementing randomization. Switch through the users' choice
In the first case let them manually insert the numbers for each player
In the second case prompt a single input per player and use the function to set all accuracies with it
In the third case display an incrementing number from 0 to 100 that resets itself once reached 100, stop the loop when the user hits ENTER key, and set all accuracies with the number that the user hit
In the default case manage error messages in case of input mistakes

Allow the user to choose between a standard 501 and an extended 701 game. Switch through the user's choice
In the first case leave the player score data member untouched (as set to 501 by default constructor)
In the second case use player score setter function to set the score of both players to 701
In default case handle error messages for input mistakes

Simulate a 50:50 game between the players to decide who will start first the real game
Make both players attempt a single bull throw and store the returned function value
Compare the two values, and set the player who scored the highest value to start game
Otherwise if both players happened to score the same, repeat the throw for both of them

Begin the real game by setting a series of nested loops that update the various counters accordingly:
Starting from 0 and incrementing by one until reaching the pre-defined number of total matches
Reset both players' number of sets won to 0
Increment the number of won matches of any player that was first to win 7 sets
While any of the two players has still not reached 7 sets won
Reset both player's number of games won to 0
Increment the number of won sets of any player that was first to win 3 games
While any of the two players has still not reached 3 games won
Switch from one player to the other, beginning from the one that won the 50:50 try
If the simulation was chosen, then both players make use of the AI system
Otherwise the first player will be the user, and the second player will use AI

When the simulation is ended, output the outcome of the entire game
    Output the number of matches each player has won
    Output a list of the set winning combinations for each player in percentages, looping through the score combinations list
Otherwise if the interactive game was chosen
    Prompt the user with the choice to end the program or to start another game

The AI system
Store the current player's score in a temporary score variable
While the currently selected player's score is bigger or equal to 2 and they haven't used all their 3 darts yet
    Loop through the double leaving doubles list
        If the score is equal to any of them, aim a double throw at the half of the dLd value
        Decrease the score by the number returned by the function
    Loop through all the dartboard numbers
        If the score is equal to a double of any of them, aim a double throw at that number
        Decrease the score by the number returned form the function
    Check if the score is equal to 50 (a bull), and if yes, aim for a bull throw, and decrease score with it
    Check if the score is bigger than 50
        Loop through each number of the dartboard and each value of the dLd vector
            Calculate the triple and the double of each number of the dartboard, individually
            Check if the score is equal to the sum of any of these and any of the dLd values
                If yes, then aim accordingly a triple or a double of the dartboard number
                Subtract the returned value of the function from the score
            Otherwise check if the score is equal to the sum of any of those values and 50
                If yes, then perform the same actions as in the previous situation
    Check if the score is bigger or equal to 62
        If yes, then target 20 and attempt a triple throw at it, decrement the score with the return no.
    Otherwise if the score is between 62 and 50, included
        Calculate the difference between the score and 50, then aim for a single on the difference
        Decrease the score with the returned value from the single throw function
    Otherwise if the score if smaller than 50
        Check if the score is 3, if yes aim for a single throw at 1 and decrease score with return no.
        Loop through all the doubles leaving doubles
            Calculate the difference of the score with the closest double
            Aim a single throw at the difference, decrease the score with the returned value

    Decrease the number of throws left
    Check if the score has reached 0, if yes, increment that player's games won and announce the winner

Reset the scores whenever one of the players has won
Otherwise if the score went under 2 and it is not 0
    Reset that player's score to the temporary score saved before his turn
    Reset his throws back to 3 and end his turn
Otherwise reset current player's number of throws
Switch players

The interactive part
Prompt the user with the choice of aim, 3 for a treble, 2 for double, 1 for single, 0 for a bull

Check their choice, if it is between 0 and 3 both included

- If their choice is 0
  - Aim for the bull and decrement their score with the return no.
  - Display to the user the outcome of their attempt
- Otherwise, prompt them to choose a number from 1 to 20 to choose from, or 25
  - If 25 was chosen, attempt a single throw at the outer bull, decrement score
- Switch through their throw type choice
  - Display what they are attempting for in each case
  - In the first case, throw a single with the chosen target, and decrement score
  - In the second case, throw a double with chosen target and decrement score
  - In the third case, throw a treble with the chosen target and decrement score
  - Display the score they hit
- Handle any input mistake and display an error message

Handle any input mistake of the throw choice and display an error message

The graphic part

Use laMothe's code for all the graphics functions and console buffer handling
Use Set_Color() function to change the color of the output messages and background
Use Draw_String() function to position the cursor at a particular position on the console by passing x and y coordinates as arguments for the function

Implement the game interface by dividing it in two parts:
Initial part handles player names' input, setting of accuracies and game type choice
Re-style all output to be displayed centered on the console
Accuracy settings and game length choice are displayed on the same position by clearing the lines and re-writing over previous text
Use Sleep() function accordingly to give a more familiar "loading" and page refresh impression

Second part displays each user's stats on the top right part of the console, one under another
Display each player's name, accuracies, number of throws left (using small graphical darts)
Darts disappear one by one whenever a player number of throws is decreased
On the top left part of the console display a "dartboard", i.e. the numbers of the dartboard in order to form a rectangle for visual check of where to throw
Display updates on throws, attempts and hit numbers for both players on the bottom half of the console by using sentences that clear previous text and write on the same lines