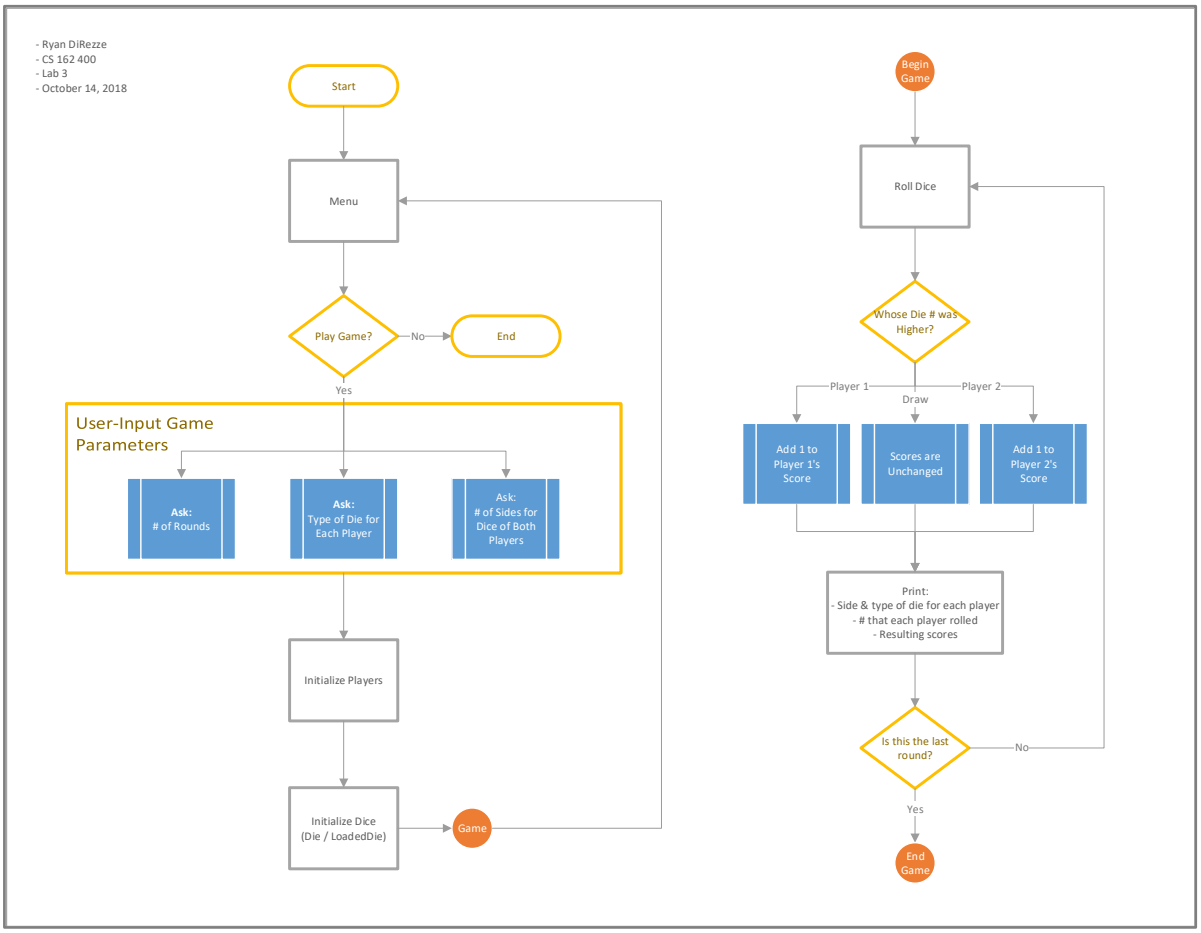
Ryan DiRezze
CS 162 400
Lab 3: War Game (with Dice)
October 14, 2018

LAB 3 (DESIGN + REFLECTION)

DESIGN / FLOWCHART

Lab 3's flowchart (PDF) is included within this assignment submission's .zip file.

Figure 1



TEST CASES

Program test table, including: test plan, expected output, and actual output can be found in this assignment's .zip file. A copy of the test table is also within this document's, *Appendix*, section as *Figure 2*.

REFLECTION

This assignment reflection will cover the following content: (1) Design changes while developing and implementing the assignment's program, (2) problems that were experienced and how those problems were overcome, and (3) what was learned from this assignment.

The first problem that I encountered was related to the inheritance of Die class attributes within the LoadedDie class. My initial constructor design with both classes output text that requested that the user input the number of desired sides for that particular die. Because of this, the LoadedDie constructor would print and accept user input twice for the number of sides on the dice. The first text statement was as if the Die constructor was initialized, then the LoadedDie constructor was initialized, replacing the Die constructor's inputs.

I overcame the above problem by requiring an integer parameter when using the Die and LoadedDie class constructors, which moved the request for user input to determine the number of sides on the dice from the constructor to outside of the constructor. By doing so, I removed the "duplicate" request for the number of sides for the LoadedDie object, and I learned about how inheritance works (at a high level) within C++.

It appears as if a "child" class that inherits attributes from a "parent" class executes the constructor of the "parent" class, then executes the constructor of the "child" class to overwrite "parent" class member variable values and/or expand upon the "parent" class' member variables. After experiencing and resolving this issue, the fact that a "child" constructor executes its "parent" constructor as well makes sense due to the structure of the "child" class' constructor code: "LoadedDie(int sides) : Die(sides)..." As shown here, the code suggests that the constructors of both the "parent" and "child" classes are executed when initializing an object with the "child" constructor and class attributes.

Beyond the above problem, no other program design changes were evaluated and implemented to overcome problems when implementing the program to satisfy the program's given requirements. All other changes were related to program aesthetics, which related to text outputs and user inputs. One example of a detailed aesthetics change was the consolidation of two separate lines of text, asking for user input for a particular game attribute (for a different "Player": Player 1 and Player 2). The change resulted in one line of text that asked the user for input for both "players", then had two input "lines" (with "cin") beneath the particular line of text that asked the user for inputs.

With Lab 3's assignment, the major challenge that was experienced and resolved related to the use of a "child" class' constructor when inheriting members from a particular "parent" class. This was resolved by moving text outputs and user inputs from within the constructor to outside the constructor.

Figure 2

Test Case	Input Value(s)	Driver Functions	Expected Outcome(s)	Observed Outcome(s)
Incorrect menu selection (special chars)	Input = %, &, *, (,), #, $	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Loops back to the question, prompting the user for input (once)	Loops back to the question, prompting the user for input (once)
Incorrect menu selection (positive integers)	Input >= 0	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Loops back to the question, prompting the user for input (once)	Loops back to the question, prompting the user for input (once)
Incorrect menu selection (negative integers)	Input < 0	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Loops back to the question, prompting the user for input (once)	Loops back to the question, prompting the user for input (once)
Incorrect menu selection (wrong letter)	Input = (letters != 'A', 'B', 'a', or 'b')	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Loops back to the question, prompting the user for input (once)	Loops back to the question, prompting the user for input (once)
Correct menu selection to begin program (uppercase char)	Input = 'A'	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Starts the program; Asks for the number of rounds	Starts the program; Asks for the number of rounds
Correct menu selection to quit program (uppercase char)	Input = 'B'	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Quits the program	Quits the program
Correct menu selection to begin program (lowercase char)	Input = 'a'	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Starts the program; Asks for the number of rounds	Starts the program; Asks for the number of rounds
Correct menu selection to quit program (lowercase char)	Input = 'b'	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Quits the program	Quits the program
Number of rounds input validation (Incorrect input type)	Input != <int>; Input = $, %, &, …; Input = a, B, c, D, …;	cin >> maxRounds; intValidation(maxRounds);	Loops back to the question, prompting the user for input (once)	Loops back to the question, prompting the user for input (once)

Number of rounds input validation (correct input type - very small)	Input = 1	cin >> maxRounds; intValidation(maxRounds);	Accepts input, then asks the user to select the type of dice for player 1 & 2 When game executes, only one round is be played	Accepts input, then asks the user to select the type of dice for player 1 & 2 When game executes, only one round is be played
Number of rounds input validation (correct input type - very large)	Input = 1,000	cin >> maxRounds; intValidation(maxRounds);	Accepts input, then asks the user to select the type of dice for player 1 & 2 When game executes, 1,000 rounds are played	Accepts input, then asks the user to select the type of dice for player 1 & 2 When game executes, 1,000 rounds are played
Wrong input (special char) for type of die	Input = #, $, %, ^, &, *	game() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Loops back to the question, prompting the user for input (once)	Loops back to the question, prompting the user for input (once)
Wrong input (integers) for type of die	Input >= 0; Input < 0;	game() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Loops back to the question, prompting the user for input (once)	Loops back to the question, prompting the user for input (once)
Correct (uppercase) input for 'Die' type of die (player 1 & 2)	Input = A	game() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Accepts inputs for players 1 & 2, then asks the user for the number of sides of each die	Accepts inputs for players 1 & 2, then asks the user for the number of sides of each die
Correct (lowercase) input for 'Die' type of die (player 1 & 2)	Input = a	game() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Accepts inputs for players 1 & 2, then asks the user for the number of sides of each die	Accepts inputs for players 1 & 2, then asks the user for the number of sides of each die
Correct (uppercase) input for 'LoadedDie' type of die (player 1 & 2)	Input = B	game() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Accepts inputs for players 1 & 2, then asks the user for the number of sides of each die	Accepts inputs for players 1 & 2, then asks the user for the number of sides of each die
Correct (lowercase) input for 'LoadedDie' type of die (player 1 & 2)	Input = b	game() if(tolower(input) == 'a') else if(tolower(input) == 'b')	Accepts inputs for players 1 & 2, then asks the user for the number of sides of each die	Accepts inputs for players 1 & 2, then asks the user for the number of sides of each die

Number of dice sides input validation (Incorrect input type)	Input != <int>; Input = $, %, &, …; Input = a, B, c, D, …;	cin >> numSides1 (or numSides2); intValidation(numSides1/2); player1/2 = new Die/LoadedDie(numSides1/2)	Loops back to the question, prompting the user for input (once)	Loops back to the question, prompting the user for input (once)
Number of dice sides input validation (correct input type - very small)	Input = 1	cin >> numSides1 (or numSides2); intValidation(numSides1/2); player1/2 = new Die/LoadedDie(numSides1/2)	Accepts inputs for dice for players 1 & 2, then begins the game, executing the number of rounds specified by the user	Accepts inputs for dice for players 1 & 2, then begins the game, executing the number of rounds specified by the user
Number of dice sides input validation (correct input type - very large)	Input = 1,000	cin >> numSides1 (or numSides2); intValidation(numSides1/2); player1/2 = new Die/LoadedDie(numSides1/2)	Accepts inputs for dice for players 1 & 2, then begins the game, executing the number of rounds specified by the user	Accepts inputs for dice for players 1 & 2, then begins the game, executing the number of rounds specified by the user

Menu repeats until player quits	N/A
	
	(No user input)
		while(selection == true) {
		menu(&selection);
		…
		Game game = Game();
		…
		}
			When the program starts, open menu. Then, open menu after each game is played until the user "quits" the program.	When the program starts, open menu. Then, open menu after each game is played until the user "quits" the program.