**Daicey – Log File**

**09/09/2016**

* Did a preliminary design and brainstorming of ideas on potential strategies for game play
* Partially implemented Dice.h, Square.h, Board.h, Player.h, Moves.h
* Dice.h needs modification in setters based on the notion that setting one face automatically sets the opposite since the sum of opposite sides is 7. Also, a strategy is needed to derive remaining sides based on a known side in the 3D model of the dice.
* Board.h needs to integrate the soldier and king dices.
* The row/column variables in Square class should be linked with the row/column variables in Dice Class to make sure update of one updates the other.
* The moves class might need a second look in the four directional roll functions.

**09/11/2016**

* Worked primarily on the Board class and created a board with dices located in respective places in respective orientation.
* Found out about the clockwise and anticlockwise nature of dices and used that to find remaining sides of a dice on being provided with 2 sides.
* SetCoordinates, SetRow, SetColumn functions inside Dice class got some further refactoring to check the validity of input parameters.
* DrawBoard function in Board class has been left halfway until a link has been established between the dice and square class.

**09/12/2016**

* Used pointers to solve the issue of linking the squares and their corresponding dices, if any.
* Some modification within the square and dice classes to integrate pointers to setup a link.
* Completed DrawBoard and UpdateBoard function in the Board class. Also, integrated a multidimensional string array that will be utilized later for serialization.
* The functions within Board class that print results to the console should be separated in a view class.
* Having issues with coming up with a proper way to access the Board class objects from other classes without compromising on Data Encapsulation.

**09/15/2016 (2.5hrs)**

* Made the GameBoard multidimensional array in Board class private and defined proper selectors/mutator functions to achieve so.
* Created a separate BoardView class and moved the display/serialization functions for the Board
* Merged the Moves class with the Player class with the intention of forming derived classes of Human/Computer later.
* Rearranged the code to meet the Project description of required classes.

**09/16/2016 (2hrs)**

* Implemented MakeAMove() function along with its supporting KeepMovingVertically() and KeepMovingLaterally() in Player.h
* Implemented a Play function in Human.h
* None of these new additions have been tested with inputs yet.

**09/17/2016 (4hr)**

* Implemented the Play() function in the Human class.
* Fixed an error of NullPtr exception by modifying the do while conditional in KeepMovingVertically() and KeepMovingLaterally() and adding breaks in switch statements of MakeAMove().
* There are some nullptr errors coming up even while making valid 90 degree moves,
* ISSUE: Moves are being made even when blockade exists on the path; also wipes the blocking dices entirely from the board while making the illegal move.
* Solved the above issues in the evening. Multiple changes in existing functions had to be made throughout the Player class.
* The destination and path validation, dice swaps works properly at this point.
* ISSUE: Any dice can capture any dice right now. Need to make it so only the opponent’s dice can be captured.

**09/18/2016 (2hr)**

* Made the functions in Player class Protected.
* Partially implemented the Game class up to the main do-while loop.
* In the evening, completed the do-while loop. The human part can be played now.
* ISSUE: If user enters char instead of int, it leads to an infinite loop. Need to validate the int input
* NEXT: Fix the existing issues, and implement the Tournament class and ErrorDisplay class. The last thing on the list are Computer’s strategy and serialization.

**09/19/2016 (4hr)**

* Can only capture opponent’s dice now. A simple if statement in the IsValidDestination() did the trick.
* Error handling done for user input of co-ordinates.
* Found and fixed the bug of human getting multiple move turns due to improper transfer of controls.
* Added the functionality to allow user to pick a path in case of a 90 degree turn. If the user chosen path is invalid, the code will automatically select the next best path. (Done by adding an optional parameter to MakeAMove()).
* Partially implemented the Notifications class and added notifications to the player class.
* Any big issue aren’t pending as of now.
* NEXT: Complete Notifications class and the tournament class.

**09/20/2016 (2.5hr)**

* Almost fully implemented the Notifications class for the classes written so far.
* Potential issue is that the input validation waits until all the coordinates and path choices have been entered. Nothing fails, it’s just weird to let user select path when the destination is out of bounds already.
* Modified code to printing board only after each user makes a valid move
* Implemented the Tournament class and integrated it with the game class properly.
* Handled the user choice to quit or continue after each round.
* Tested thoroughly to make sure the tournament exits/continues as the user wants
* At the point, the game seems to be have all the essential components implemented for it to be a human game.
* NEXT: Serialization and Computer Strategy

**09/21/2016 (3hr)**

* Implemented first half of serialization in BoardView class where the program asks the user their wish to serialize in every step along the game. Writes to file and exits the game if user wishes to.
* Found bug – If user entered more parameters than necessary, they were being carried over to the next cin request causing discrepancies in values assignment to variables.
* Fix: Added Cin.ignore() throughout the cins sections of code.
* Next Step: Restoring the serialization file to the gameboard in the beginning of a game

**09/22/2016 (2.5hr)**

* Moved the serialization components to a separate Serializer class which made things simpler.
* Wrote a function to read the gameboard from the text file and restore to the SerializedGameBoard string array.
* Attempted to use Regular expression to read the scores and next player, but it wouldn’t work for some reason.
* The regex\_match() is unable to find any matches at all, so it is probably something wrong with the implementation itself rather than the expressions.
* Next step: Get the regex to work so that the values can be restored to the respective variables. Then, the respective flags in the board, squares and dice can be set based on the retrieved values from file.

**09/25/2016 (5hr)**

* Have a decently working restoration portion of the serialization.
* Issue 1 (Potentially Fixed): King is fixated in [4] position, so the restoration needs to be done carefully to not place any other die in that index
* Issue 2: Need to come up with a formula to calculate the front and rear sides of the die during restoration.
* Combined serialization with the tournament overall and setup, have a full human side of the game.
* Issue: The reading from file itself seems to be fine, however, there seems to be some issue while setting the flags. As a result, while printing the game board from the tournament, some of the dice show incorrect values for top-right.

**09/26/2016 (1hr)**

* Fixed the issue associated with restoring serialization file. Turns out the botCount was being incremented instead of a humanCount in one part of the code which messed up the indexes being updated.
* At this point, the game has all the features for the human to play. Need to implement the computer strategy next.

**09/27/2016**

* Modified the function to set the beginning orientation of dice in order to accommodate the differences for human and computer home row dices.
* Changed the BoardView to display Top-left face values for computer dice since the right side is actually left side for user viewing from other side.
* Implemented the Play function in Computer class to capture opponent king if possible, and capture hostile dice if threat detected for computer’s King.
* Partially implemented the part to capture other ordinary opponent dice but it still has an issue
* Issue: RunningOverOwnDice error keeps getting displayed while computer is checking possible attack modes to capture opponent dice. It is abnormal cause the destination at hand is always an opponent die, so need question of capturing own die at this point.
* Issue: Also, noticed that one of the times the opponent die wasn’t captured though it was possible to make that move.
* Next: Fix issues while capturing opponent dice, Implement blocking while under threat, and make it possible to move the king. Also, come up with a recursion to make a proper move if no captures are possible.

**09/29/2016 (3.5hr)**

* Implemented blocking while under threat and made it possible to move the king.
* Potential Issue: Once there was a weird situation where the blocking move was made even when the hostile die with a top value of 1 was right next to the king. Didn’t occur later when I commented out the blocking code itself though. It was weird.
* Apart from that, the game ran quite smoothly.
* NEXT: Implement Breadth First search to find normal moves when no threats/captures are imminent.

**09/30/2016 (4.5hr)**

* Added the function to calculate the remaining faces during serialization.
* Also fixed the bug associated with accidentally setting king to captured while there are less number of dices in the restore file.
* Implemented the best move search part in Computer’s play() function if no defenses, captures are possible.
* Fixed the “blocking in wrong spot” issue noticed yesterday by making the correct function call in the TryBlockingHostileDice function.
* Added print statements to announce the winner at the end of the tournament.
* At this point, the game is basically complete to play Human vs. Computer.
* Next: Implement Help for Human and refractor the code based on the rubric.

**09/1/2016 (2hr)**

* Worked on notifications display to ensure error printing isn’t done when computer is running its’ play algorithm.
* Added functions in notification class to display computer’s thought process to the user.