5/1/2020
CPSC 224
Bruce Worobec

Group Project Final Report

Kevin Lunden, Nicole Bien, Eric Gustin, Jackson Lindsay

Our group went in the direction of a modified version of the Lizard-Spock-Yahtzee project that we worked on for homework four. We created a game dubbed “Rainbow Yahtzee” that uses colors instead of numbers for the die faces and also added multiplayer capabilities.

We began with the rules, creating a game that would have many of the same rules as Yahtzee so that anyone familiar with the original could easily pick up our modified version and play. We changed wording to reflect the color aspect, we modified straights to become color pallets and of course we changed the Yahtzee option to Rainbow. Our game is slightly more difficult than regular Yahtzee due to the fact that we have seven colors required to meet the Rainbow criteria compared to six die faces for a normal Yahtzee.

We then began crafting the functional requirements. We inserted provisions for the user to be able to customize the game to their liking prior to playing in addition to keeping many of the same functional requirements for a regular game of Yahtzee. We didn’t know at the time that we would be adding multiplayer capabilities, so that wording is absent from our original functional requirements deliverable.

Creating a project plan was the next step. We found that on our first pass through, we didn’t have nearly enough steps and that each step was going to take far too long to complete (well over the 4 hours recommended per task). We were able to divide each of those steps into substeps to create more attainable goals. Then, after looking over examples given in class, we

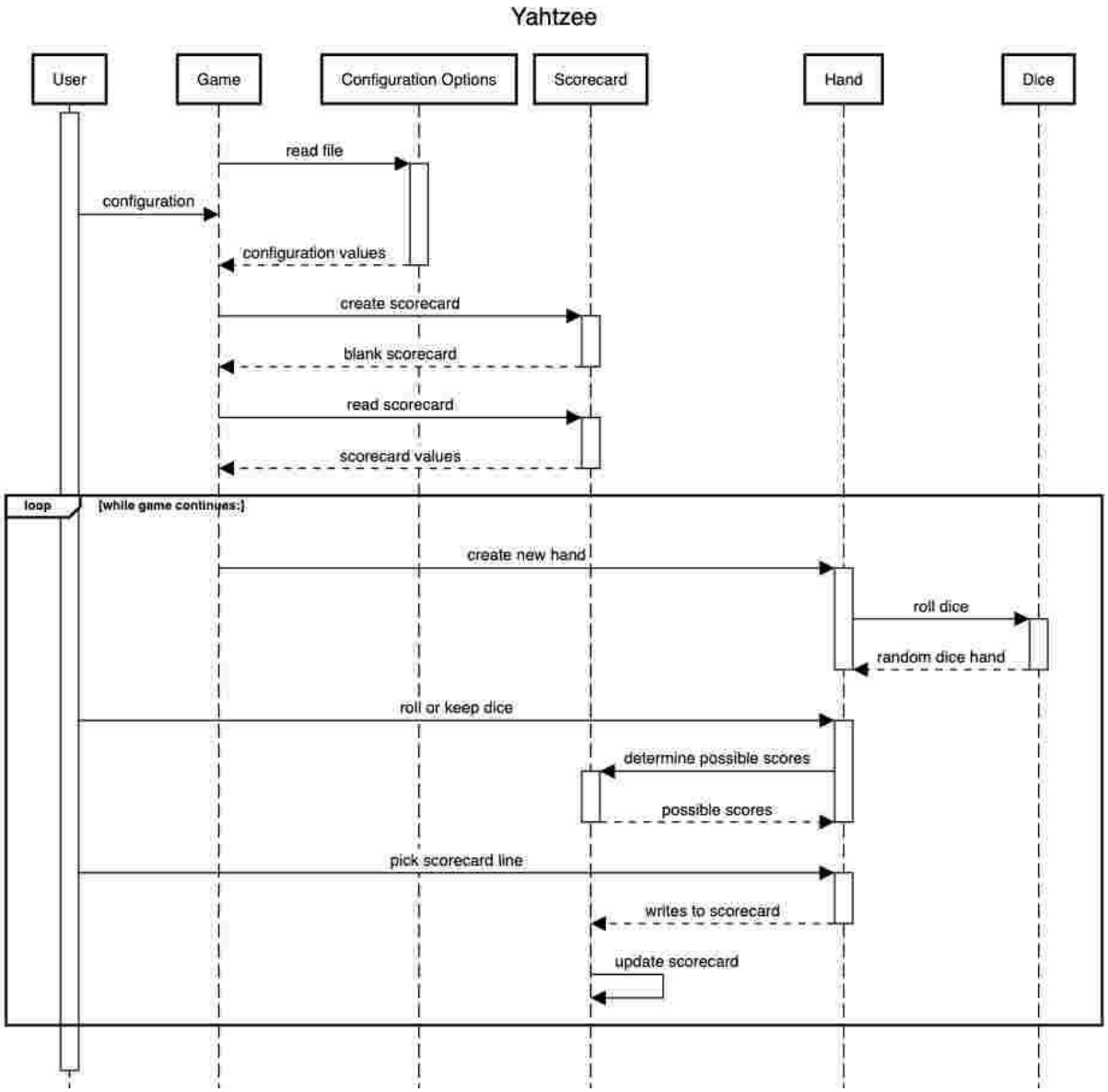
were able to come up with even more steps for our project plan that would ensure a successful group project with easily delegated tasks.

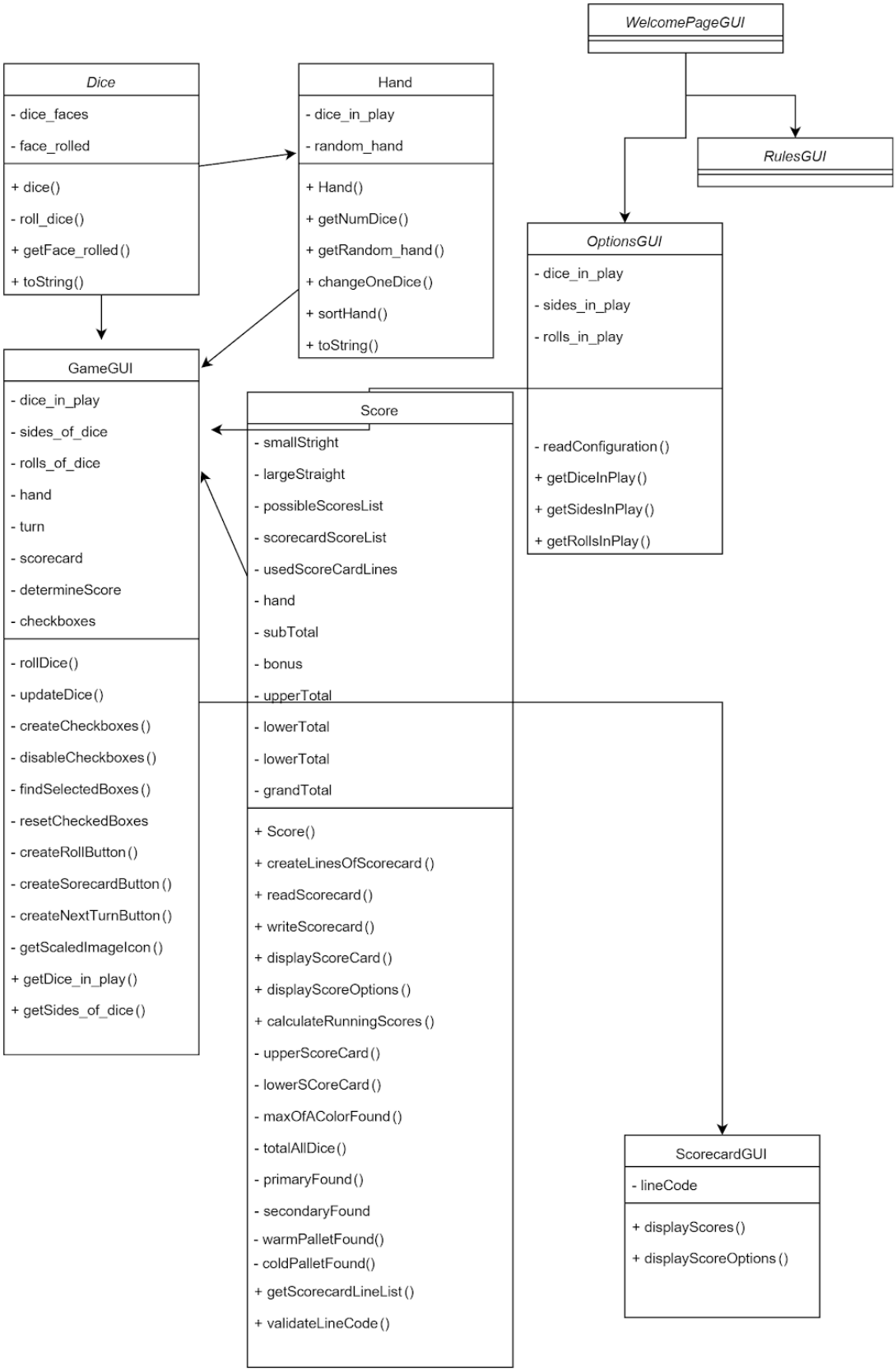
Finally, we created our system test plan. This was more difficult than we thought it would be. Going through the functional requirements to gather the necessary steps was easy. Going through and creating test cases for each step was not. We attempted to think of every possible combination and permutation that might allow a user to break our program. This is something that is very hard to think of off the top of your head. We found that playing our game multiple times through with the intention of trying to break it was a useful exercise to discover other possible holes in our code. We would play games in every way that we imagined a user might interact with it and sought out to create testing steps that would ensure that all possible scenarios would result in a satisfying user experience. Through multiple rounds of system testing we managed to make our program almost completely airtight to users attempting to break it.

What worked: The ability of the team to modify pre existing code to fit our new project requirements. Being able to reuse code that we had already created allowed us to focus more on the UI and making a flawless interactive experience for the players. Our group managed a successful rework of the game of yahtzee that is simple enough for users that are familiar with yahtzee to play without struggling but also new enough to add an exciting amount of twist to the original game.

What didn't work: Group interaction with GitHub was a struggle. We were frequently pulling and merging and messing up our repository history. This caused a few headaches and some time forcing changes to GitHub. Though using pre-existing code was useful, it was all challenging as some group members had to quickly familiarize themselves with the code.

What we would do differently: Upon reflection, we got caught up a few times having procrastinated a deliverable and having to work more closer to the due date. This ended up working out for our team, but it is possibly not the best practice and something that we would like to change in the future given another group assignment of this nature.




WelcomePageGUI
Dice
- dice_faces
- face_rolled
+ dice()
- roll_dice()
+ getFace_rolled()
+ toString()
Hand
- dice_in_play
- random_hand
+ Hand()
+ getNumDice()
+ getRandom_hand()
+ changeOneDice()
+ sortHand()
+ toString()
RulesGUI
OptionsGUI
- dice_in_play
- sides_in_play
- rolls_in_play
- readConfiguration()
+ getDiceInPlay()
+ getSidesInPlay()
+ getRollsInPlay()
GameGUI
- dice_in_play
- sides_of_dice
- rolls_of_dice
- hand
- turn
- scorecard
- determineScore
- checkboxes
- rollDice()
- updateDice()
- createCheckboxes()
- disableCheckboxes()
- findSelectedBoxes()
- resetCheckedBoxes
- createRollButton()
- createSorecardButton()
- createNextTurnButton()
- getScaledImageIcon()
+ getDice_in_play()
+ getSides_of_dice()
Score
- smallStright
- largeStraight
- possibleScoresList
- scorecardScoreList
- usedScoreCardLines
- hand
- subTotal
- bonus
- upperTotal
- lowerTotal
- lowerTotal
- grandTotal
+ Score()
+ createLinesOfScorecard()
+ readScorecard()
+ writeScorecard()
+ displayScoreCard()
+ displayScoreOptions()
+ calculateRunningScores()
- upperScoreCard()
- lowerSCoreCard()
- maxOfAColorFound()
- totalAllDice()
- primaryFound()
- secondaryFound
- warmPalletFound()
- coldPalletFound()
+ getScorecardLineList()
+ validateLineCode()
ScorecardGUI
- lineCode
+ displayScores()
+ displayScoreOptions()