Implementation

Link to requirements

Four player support

The majority of the changes to the code made to implement support for up to four players, as shown in requirement #15, were made in the RoboticonQuest class, as this is the main controller for the game. The nextPlayer method was modified so that it looped through the playerList instead of switching between players, which therefore made it possible to add more than two players to the turn cycle. getWinner was replaced with 3 new methods: getPlayerWithHighestScore, playerToName and getWinnerText. This was done to increase separation of concerns and to allow for the easier implementation of more players, as the single method would have become cluttered when attempting this and would be much less readable. The reset method was modified to add the correct number and type of players to the list of active players; playerList. implementPhase was renamed to runCurrentPhase to to make its purpose more clear and prevent confusion.

Outside of the RoboticonQuest class, the main changes to code were made in GameScreen in order to start the game with the correct names and types of players. Within GameScreen, newGame was modified to take two parameters for the numbers of human and AI players, instead of a single boolean value for if the game was against AI or not. These are then passed into the updated reset method within RoboticonQuest.

The GUI was modified in a few places. Firstly, the start screen was changed to allow the player to individually select the number of human and AI players, only allowing up to 4 players and no less than 1 human player (as required by requirement #16) to be selected. The end screen was also changed to show the scores for the correct number of players.

Chancellor minigame

To implement requirement #49, the “capture the chancellor” minigame, a Chancellor class was added to control the event. This class manages the instantiation and positioning of the chancellor within the event. A chancellor effect was added to PlayerEffectSource, therefore allowing the minigame to be started as an event. The Player class was modified to keep track of how many times each player has caught the chancellor, so that the points from doing so can be added to the score in the calculateScore method, as required by requirement #51. GameScreenActors was changed to manage the GUI elements of the chancellor event to make sure it is correctly positioned and displayed.Finally, changes were made to GameScreen in order to implement the logic for the chancellor event. GameScreen is also responsible for the 15 second time limit to catch the chancellor for requirement #50.

Extra additions

We created extra tests in order to make our development easier, as we would know we were not breaking the parts of the game that already existed. We also decided to use exceptions instead of error codes used by the previous team. The error codes sometimes used boolean false for an error and true for a success, or used an enum with values for error or success. This is an issue as it can create unexpected behaviour if someone was to forget to check for an error code. Another reason for this change is that it makes debugging much easier, therefore making the development of other features much easier.

We also changed some aesthetic features to improve the overall look of the game. We replaced the old team's Roboticon sprites with our own, as we felt they had a much more appealing aesthetic and fitted the theme of the game better than the old ones. The start screen was also updated to use our own game name and Roboticons. We felt it was important to improve the looks of the start screen as it is the first thing people will see when playing the game.