Project 1

Title:

Sorry Board Game

Course:

CIS-17C

Section:

43400

Due Date:

May 9, 2021

Author:

Drake Fafard

Sorry Explained[1]:

https://www.ultraboardgames.com/sorry/game-rules.php

Setup

In the game, there are 4 players of a unique color, each of which has 3 pawns that they try to put in home. Each of these pawns starts in their color's starting zone on their color's side. As previously mentioned, the goal is to get all 3 pawns into home first as this causes that player to win.

Moving

Any forward moving card can be used to move a pawn out of start. Doing so will cause the pawn to move that number of moves forward, with the first move being the move from start to the board. The pawns move clockwise along the board, and landing on a slide start's beginning (indicated with ->) at the end of the turn will cause that pawn to slide down the slide to the end of the slide. When a pawn uses a slide, it knocks down any pawns along the path of the slide including pawns of their own color. However, pawns cannot use slides on their color's side. When moving normally, pawns will jump over other pawns and leave them unaffected. At the end of a move, a pawn will bump off any pawn that is on that final space. Safety zones are filled with ** and proceed the Home, and they prohibit targeted swaps for pawns within the zone. A pawn cannot pass the entrance of its color's safety zone, instead they use that entrance to enter the safety zone. However, a pawn may use a backwards move to exit the safety zone.

Home

To reach home, the pawn must land in home with no movement left, otherwise the move is invalid.

Cards

(1) Move one of your pawns forward one space.

(2) Move one of your pawns forward two spaces.

Then draw again.

(3) Move one of your pawns forward three spaces.

(4) Move one of your pawns backward four spaces.

(5) Move one of your pawns forward five spaces.

(7) Move one of your pawns forward seven spaces or split the forward move between two of your pawns

[1] https://www.ultraboardgames.com/sorry/game-rules.php

- If you use part of the seven to get a pawn home, you must be able to use the balance

  of the move for another pawn.

(8) Move one of your pawns forward eight spaces.

(10) Move one of your pawns forward ten spaces or move one of your pawns backward one space.

(11) Move one of your pawns forward 11 spaces or switch any one of your pawns with an opponent's.

- You may forfeit your move if you do not wish to change places and it is impossible to go

  forward 11 spaces.

- You may only switch pawns in play on the open track, not at start, home, or in a safety zone.

- If your switch landed you on a triangle at the beginning of a slide that shows your color,

  slide to the end.

(12) Move one of your pawns forward 12 spaces.

(Sorry) Move a pawn from your start area to take the place of another player's pawn, which must

return to its own start area. Or move one of your pawns forward four spaces.

- If there is no pawn on your start or no opponent's pawn is on any space you can move to and

you cannot move any of your pawns four spaces, you forfeit your turn.

Program Composition:

The program contains 2186 lines of code, 5 classes, and a main file.

The main file contains an object to the menu class. The main file's purpose is to construct the menu object.

The board class contains 12 functions, 1 map, 1 map iterator, and 8 sets. It contains the board which holds the pawns and manages movement across the board. It holds functions which can swap, move forward, and move backwards across the board and relies on the game class to utilize the calling of these functions based on the cards.

The card class contains 10 functions, 1 map, 1 dynamic array, 1 stack, 1 tree, and 1 recursive sort. It manages drawing the cards and stores information on each card, which the menu class uses in displaying the rules.

The game class contains 13 functions, 2 maps, 1 list, 1 queue, and 2 hash functions. It manages the player turns, the functionality of the sorry cards, the position of the pawns, and the player inputs.

The menu class contains 3 functions and a game object. It manages multiple game sessions and displays the rules of the game.

The CardTree class contains 6 functions and a CardTree object array. It serves as a node which can link to other nodes of its same class, creating a Tree. The tree is used in managing card display.

Program Functionality:

The program starts by creating a menu object, which will then start with displaying the rules of the game and asking the user if they want to start a game of Sorry. This second part will run until the player no longer wishes to start a game, allowing multiple games to be run in one session. If they start the game, then the game class will ask the players to input their names. Upon doing so, the game will begin.

The game works by calling player turns until one of them wins, and the turn will start by moving to the next player, displaying the board, and drawing a card for the player. The display of the card and drawn card depends on the card class's draw and the card tree stored in the card class. Depending on whether any moves can be done with the card, the player will be able to choose a pawn to move. The game class will send information to the board class functions based on the players choice and the card that was drawn and will rely on the function's feedback to determine if the player's choice was valid. If it was, then it will once again call on the function with the information that this is a real move instead of a test move. The board class will then update the board with the new move, and the game class will update their own quick access pawn map with that information.

Pseudocode for Board::forwardMove:

```
Receive in parameters variables for pawn, position, whether the move is real, and how many moves to take

Declare variables for new color, new position, eliminated pawn

Declare variable for original position, set this equal to position

If the pawn or position is an invalid format, return false

If the move number is 0 return true

If the position is a home spot return false

Loop for the move number {
```

```
If the position is a start spot {
    Set new position's color equal to the pawn's color, then slide 3
    If this is a real move, and if it is the final move modify the board appearance, bump any pawn on this spot
} else if the position is a normal spot {
    If the position is the first normal spot {
        Move to the first slide of that same color
        Perform a slide if it is the last move
    }
    Else If the position is the 5th normal spot {
        Move to the fifth slide of that same color
        Perform a slide if it is the last move
    }
    Else If the position is the 6th normal spot {
        Transition to the next color
    }
    Else {
        Move to the next normal spot
    }
} else if the position is a slide spot {
    If this is the end of a slide, move to the next normal spot following it
    Else if the position is on slide 2, move into the safety zone if this is the color's safety zone
    Else, move to the next slide spot
} Else if the position is a safety zone {
    If it is the last safety zone spot, it must be the end of the move otherwise the move is invalid
    Else, move forward to the next safety zone spot
```

```
        }
    }
```