WHEEL OF FORTUNE

PRESENTED BY

RADIAH JONES

11/07/2019

About Wheel of Fortune

Wheel of Fortune is a television game show that debuted in 1975 and now has over sixty international adaptations. Three contestants attempt to solve word puzzles to win cash and prizes determined by spinning a giant wheel.
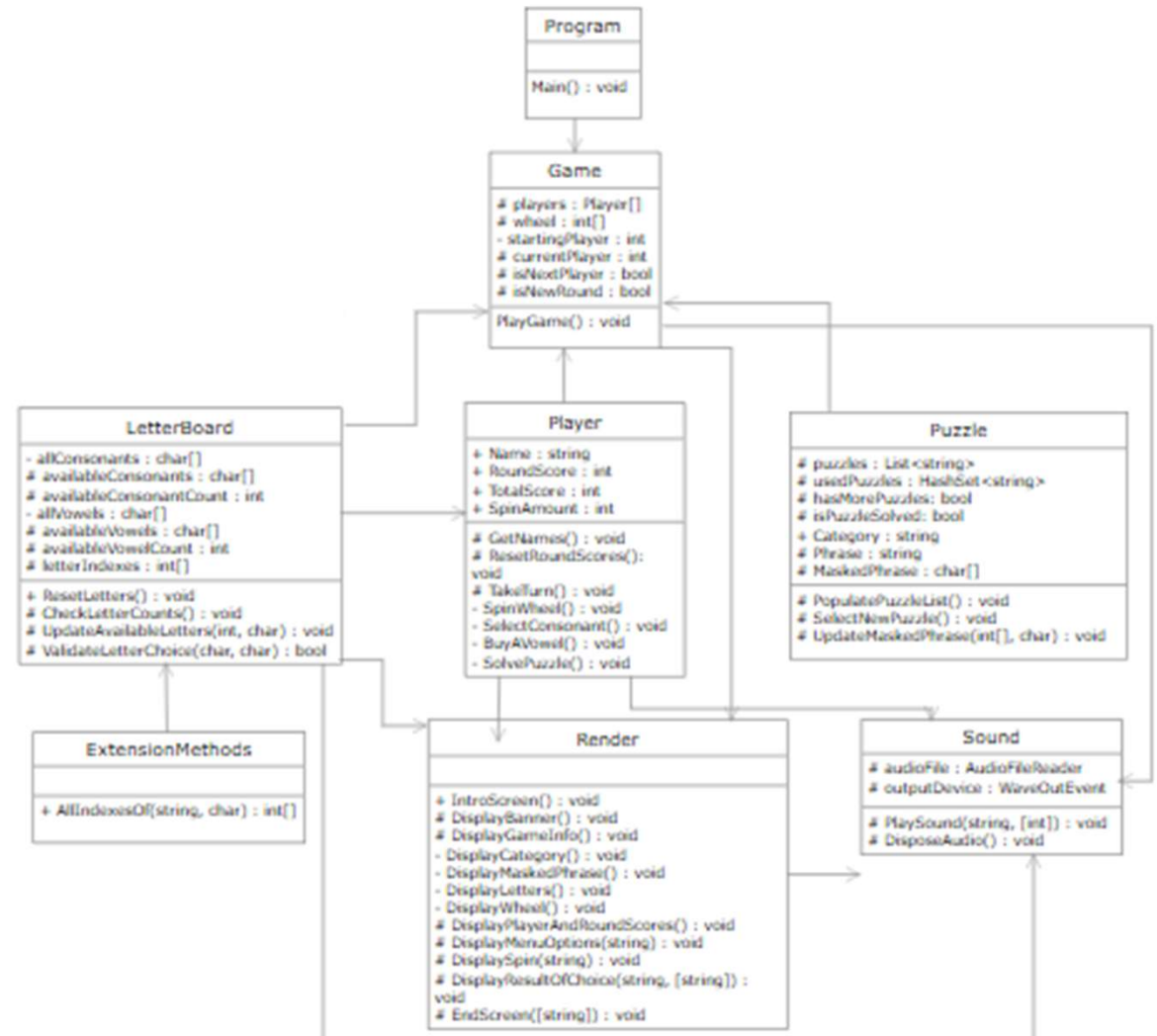
Rules of the Game:

Several successive rounds constitute one entire game. During a player's turn, they have 3 choices:

1. **Spin the wheel** - to get a dollar amount and guess a consonant. For each consonant guessed correctly, they accumulate the multiple of that dollar amount. If the consonant they guess is not in the puzzle (or has already been selected), play passes to the next person. If the wheel element chosen is Lose-a-Turn, the player's turn ends. If the wheel element chosen is Bankrupt, the accumulated total is reset to 0 and the player's turn ends.

2. **Buy a vowel** – the player does not spin. The play must pay $250 to guess a vowel and have all occurrences of that vowel shown in the puzzle. If the vowel they guess is not in the puzzle, play passes to the next person.

3. **Solve the puzzle** – During a player's turn, if they think they can solve the puzzle, they input their guess. If it is correct, they win the money they have accumulated during the round. Only the player who solves the puzzles gets to keep the money accumulated in a round. If the guess is incorrect, play passes to the next person.

Game Flow

Player

Spin
- **$$**
  - **Guess**
    - **$$, Reveal, Update, Play Again**
    - **Next Player**
- **Lose Turn**
  - **Next Player**
- **Bankrupt**
  - **Next Player & $0**

Buy Vowel
- **- $250 & Guess**
  - **Reveal, Update, Play Again**
  - **Next Player**
- **Next Player**

Solve
- **$$ to Total**
  - **Reveal, Play Again?**
    - **New Puzzle Next Player**
    - **Display totals and End**
- **Next Player**

UML

Program

Main() : void

Game

players : Player[]
wheel : int[]
- startingPlayer : int
currentPlayer : int
isNextPlayer : bool
isNewRound : bool

PlayGame() : void

LetterBoard

- allConsonants : char[]
availableConsonants : char[]
availableConsonantCount : int
- allVowels : char[]
availableVowels : char[]
availableVowelCount : int
letterIndexes : int[]

+ ResetLetters() : void
# CheckLetterCounts() : void
# UpdateAvailableLetters(int, char) : void
# ValidateLetterChoice(char, char) : bool

Player

+ Name : string
+ RoundScore : int
+ TotalScore : int
+ SpinAmount : int

GetNames() : void
ResetRoundScores() : void
TakeTurn() : void
- SpinWheel() : void
- SelectConsonant() : void
- BuyAVowel() : void
- SolvePuzzle() : void

Puzzle

puzzles : List<string>
usedPuzzles : HashSet<string>
hasMorePuzzles: bool
isPuzzleSolved: bool
+ Category : string
# Phrase : string
# MaskedPhrase : char[]

PopulatePuzzleList() : void
SelectNewPuzzle() : void
UpdateMaskedPhrase(int[], char) : void

ExtensionMethods

+ AllIndexesOf(string, char) : int[]

Render

+ IntroScreen() : void
# DisplayBanner() : void
# DisplayGameInfo() : void
- DisplayCategory() : void
- DisplayMaskedPhrase() : void
- DisplayLetters() : void
- DisplayWheel() : void
DisplayPlayerAndRoundScores() : void
DisplayMenuOptions(string) : void
DisplaySpin(string) : void
DisplayResultOfChoice(string, [string]) : void
EndScreen([string]) : void

Sound

audioFile : AudioFileReader
outputDevice : WaveOutEvent

PlaySound(string, [int]) : void
DisposeAudio() : void

GAME LOGIC

```csharp
public static void PlayGame()
{
    // display these once
    Render.IntroScreen();
    if (isNewRound)
    {
        Puzzle.PopulatePuzzleList();
        Player.GetNames();
    }
    while (isNewRound)                              // new puzzle
    {
        Render.DisplayBanner();
        Puzzle.SelectNewPuzzle();
        Letterboard.ResetLetters();
        Player.ResetRoundScores();
        Puzzle.isPuzzleSolved = false;
        currentPlayer = startPlayering;

        while (!Puzzle.isPuzzleSolved)             // new guess
        {
            Render.DisplayGameInfo();
            players[currentPlayer].TakeTurn();
            if (isNextPlayer)
                { currentPlayer = (currentPlayer + 1) % 3; }
        }
        startPlayering = (startPlayering + 1) % 3;

        // if yes, check for more puzzles. If more puzzles, isNewRound is true, else false
        Console.Write(" Play Again? Y / N: ");
        isNewRound = Console.ReadKey().Key == ConsoleKey.Y ? Puzzle.hasMorePuzzles ? true : false : false;
        Console.WriteLine();
    }

    // if ending because of out puzzles, send message
    Render.EndScreen(Puzzle.hasMorePuzzles ? "" : "********* NO MORE PUZZLES *********");
    Sound.PlaySound("themeMusic.mp4", 10000);

    // clean up unmanaged audio resources
    Sound.DisposeAudio();
}
```
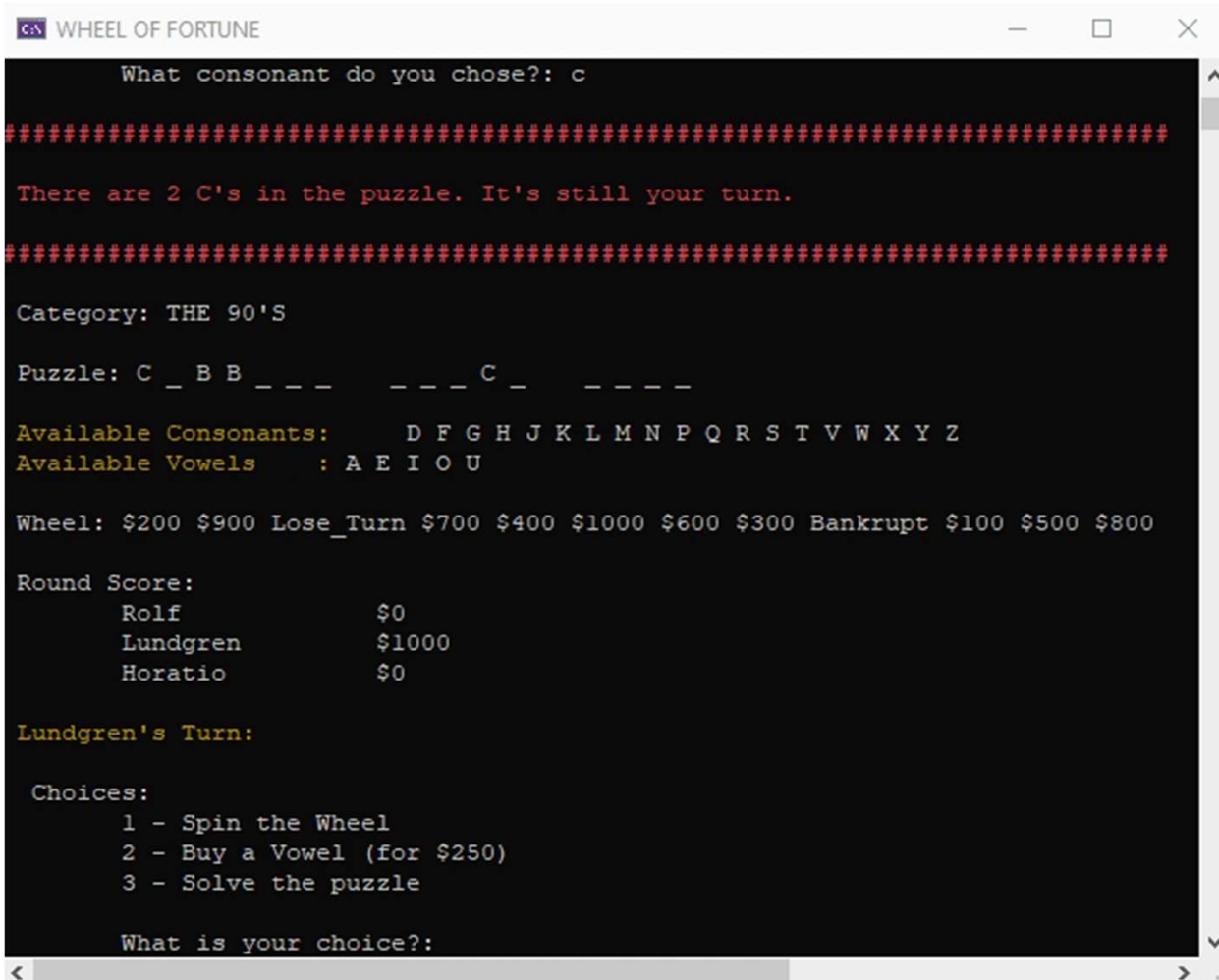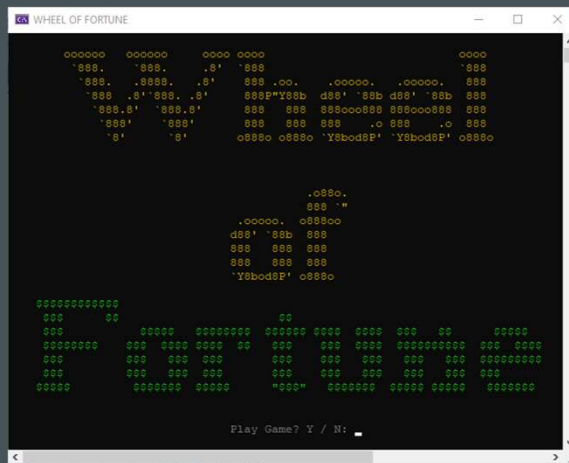
DEMONSTRATION

POSSIBLE UPGRADES

- Instructions
- Multi-player input
- Computer Player
- Time players for each turn
- Bonus Round for player with highest Total Score
- Create a Test file
- Make extension method generic
- Handle puzzles with special characters
- Connect to API with unlimited puzzles

RESOURCES

- Wheel image: https://blog.margaritaville.com/2018/10/wheel-of-fortune-home-sweet-home/

- ASCII art: http://www.network-science.de/ascii/ (Thanks to Melissa Stock)

- About Wheel of Fortune: https://en.wikipedia.org/wiki/Wheel_of_Fortune_(American_game_show)

- Extension Method: https://stackoverflow.com/questions/10443461/c-sharp-array-findallindexof-which-findall-indexof

- Sounds: https://www.soundboard.com/sb/WheeloffortuneFX

- Puzzles: https://wheeloffortuneanswer.com/

- UML Diagram Website & PowerPoint Template: https://www.draw.io/ (Thanks to Evan Slaton)