MONOPOLY - electronic version

User Documentation

1. **Specification Breakdown**

    a. **About project**

    Monopoly gives you a chance to play the favourite game that you used to have to carry around electronically, meaning you only need your computer to play it. The players can play in a hot-seat mode meaning, that you are going to have to be physically present to play the game as the network version is not (yet) implemented.

    For further information see Specification.pdf file.

    b. **Functional requirements**

    The application simulates a game of monopoly. This is a fairly simple structure, but the added value is a smarter-than-stupid AI player functionality, which decides dynamically, not very predictably with a partly randomised decision-making logics to make the game more interesting, unpredictable with possibly shocking outcomes.

    The player is going to be, however, able to avoid playing with the AI random players by populating the player slots with human players. The game is to be saveable as a Monopoly game can last for as long as several days of gametime.

2. **Architecture and Design**

    a. **High-level stages of program**

    The program is split into 4 stages:

    1) **Main menu:** where the user is introduced into the game having 4 options: New game, Load game, Help (5th stage) or Exit

        Thanks to this the game is to be considered a game - if you were not able to launch it from the main menu, there would be no game. Also, the ability of loading a saved game is necessary to fulfill the requirement of being able to save the game.

    2) **New Game Dialog:** after deciding to create a new game, the player has to choose the number of players and the amount of money each player has at the start of the game.

        Thanks to this, player can alter the game settings which is vital to having a good game.

    3) **Player Choosing:** The dialog where the player can set whether the player is an AI or a human and also set the players’ names.

This directly maps to the functionality of having option to avoid playing with the AI.

4) **Game:** The most important stage of the program where the actual gameplay takes place and the main logic is performed. Without this, the program would just be a pile of controls without any real purpose.

   This, of course, maps to the functionality of being able to play with an unpredictable AI player and it is also here where the game can be saved.

b. Functions and procedures

The whole program is event-driven and therefore the majority of functions and procedures is triggered upon an event in the graphical window. Vast majority of all functions and procedures (just F&P) are linked to events in the main Game Window, where the game is played. There are basically two groups: graphical and logical.

Graphical F&P take care of updating the game window and providing the user with relevant and correct information so that he can play without missing any information, always being clear about what he is required to do or asked about.

Logical F&P are mostly data-manipulating procedures, that are triggered after an event is fired. These F&P analyse the game state, perform data manipulation (assigning ownership, adding/subtracting money,...) and it is these F&P that's results the player doesn't see unless a corresponding graphical function or procedure displays its result.

c. Data structures

There are two main structures in the program: cards and players.

i. Cards

There are 5 types of a card, divided into 2 groups.

Purchasable cards: Purchasable cards can be **Property, Bonus or Agency cards**. They are more or less implemented in the same way.
They are basically just data containers, so that we can address a group of data (a card) easily. They all possess attributes such as name, cost, payment values, mortgage value or boolean 'mortgaged' attribute.
Without these cards the game itself would not make sense, since Monopoly is mostly about buying, selling and trading.

Special cards: These cards are played during the game. These cards are **Treasure cards and Risk Cards**. They possess some description text information as well as instructions for performing the result described in the description - being it blocking a player or adding/subtracting money.

ii. **Players**
Players are divided between AI and Human players. The difference between them is, that when there is an action required, AI Player is automatically deciding (using AIDecider class), while Human player is left with options in the graphical window and further action is triggered after clicking on a button.

d. **Global variables**
The program does not rely on global variables. Instead, all important data are assigned to either data structures described above.

e. **Algorithms**

i. The major algorithmic problem of the program was to tackle the gameflow. It is done using the GameState attribute of the game controller - Game class - which analyses the state of the game and based on the event triggered, it decides what can be done next.

ii. Another difficult part is the AI decision-making. There is no universal way on how to describe the typical behaviour during the game. Instead, the AI Players use a ‘random element’ value in every decision making - be it deciding where to buy or upgrade a card, or deciding whether to trade or mortgage a property. This ‘dangeFactor’ is generated randomly when the AIPlayer’s constructor is invoked.

iii. Other than these, the rest of the decision-making of program is performed using some kind of if-else branches or using switch.

f. **User input processing**
The user input is processed very simply. Before the game starts, there are just handlers performing the actions they have to after the corresponding button is clicked on.

During the game, there are just a few types of input that can be input. The 3 game buttons are the main tool of communication between the program and the user. This is done using a handler in the controller of the game. The game, having several game states, knows in which context the button was pressed and what it has to do with it.

When sending a trade offer, the controller (Game class) can get the selected items in the ListView Control and also the value of TrackBar where player sets the amount of offered money.

The Main Menu Button and Exit Button are analogically connected using handlers.

3. Technical Documentation

For detailed technical documentation, see ‘generatedDocumentation’ folder in the ‘Documentation’ folder of this program.

Author: Andrej Jurčo
Last Update: 30.7.2018