Documentation

Author: Bartosz Pietrzak

Target and description of the project:

The project involves programming the sokoban game. In this game, the player can move his character horizontally or vertically, diagonal movement is impossible. The player's task is to move all the boxes on the board to the switches. Boxes can only be pushed, they can only move to empty spaces and spaces with switches. When all switches have boxes on them, the level is completed. The player can also move around empty tiles and buttons. Each board must have the same number of buttons as boxes.

Description of the programme:

The program has been divided into several separate files. The main part of the program contains a gui file. It contains the SokobanWindow class responsible for implementing the graphical interface - initializing it and then refreshing the game board based on the player's movement. Moreover, this class allows you to reset a given level or load a custom level. SokobanWindow also captures keyboard input and passes it to the GameManager object to perform the move. The same file also contains classes responsible for pop-up windows with information about starting the game, moving to the next level and ending the entire game. What’s more, at the end of the game window, the user can choose whether he wants to end the game or start it from the beginning. The next file is ui_sokoban containing the main interface design created using the designer. The next class is GameManager, located in the gameManager. This class is responsible for moving the player, it considers all possible cases of movement and boxes. In order to optimize the game, GameManager also stores tiles that have changed with a given move by the player, so that the graphical interface does not have to update the entire board. The classes file contains classes representing tiles. The Tile class is responsible for storing the color of the tile, and is inherited by EmptyTile, Switch, Wall and TileThatCanBeOnSwitch.The last class allows the game to detect whether a tile is currently on a button and gives the tile the ability to change color after touching the switch. Box and Player are inherited from it. The final classes differ in their string representation. The file also contains a class selector function responsible for selecting a class based on its text representation. The next file is levelLoader. Its main function is loadLevel, which loads a level from a JSON file based on the previously provided path. Additionally, thanks to the boardCorrectionValidation function, the loaded level is checked for having the correct number of buttons and boxes. The errors file contains all additional exceptions. Each exception prints an appropriate message to the console and displays a window with a short description of the error. In addition, the program also contains a test file (navigation and loading

Documentation

tests) and 3 JSON files containing basic levels, each subsequent one becoming more and more difficult.

Instruction manual:

To run the game, the user should install all dependencies in the requirements.txt file and then run the gui.py file using the python3 gui.py command. In order for the program to function properly, the user must use an environment with a graphical interface. After starting the programme a message window will show. It contains short instructions and rules explaining what to do and how to play the game. After closing it the main game window will show up. On the right side user will be able to see a short explanation of colors of the tiles and summary of rules. Game can be restarted using the level menu in the top left corner or keyboard shortcut ctrl+R. Custom level can be loaded using either the level menu or ctrl+L shortcut. User commands player using WSAD keys, they correspond to move up, down, left and right. Player needs to push all of the boxes (yellow tiles) to the switches which are marked as red tiles. After doing so another message window will show up, informing the player that they finished the level. User needs to accept it in order to progress further. After completing all levels or a custom level message will pop up, which will give the user the option to restart the game.
If a user wants to create a custom level, they will need to create a JSON file containing a list. In that list the user needs another list, where each list corresponds to one row on the board. In those lists there need to be key words in quotation marks, representing different types of tiles. Those keywords are: “emptyTile”, “box”, “player”, “switch” and “wall”. User needs to make sure that their board contains the same number of boxes and switches and that they placed only one player. Finally all of the outer tiles must be marked as wall, this requirement was made to make sure that player can’t “escape from the board”

Reflexion:

What was done:

- Graphical interface which shows the board and additional information was created
- Smaller windows informational windows were added
- Possible errors were tested and corresponding pop up messages were added to inform user about them
- Player was given an ability to move
- Test were created to make sure that everything worked correctly during development
- Options to restart level or load a custom one were added

What wasn’t done:

- Images weren’t added to tiles, there was not enough time to create custom graphics for them
- More levels weren’t created since it was time consuming and made GUI testing longer

Unexpected obstacles:

- GUI creation was harder and more time consuming than anticipated
- During level creation process I could find more and more potential problems that could made it impossible to play or complete, so there needed to be additional test and custom issues created

Documentation

- It was hard to make the movement work correctly, there were several issues with moving the box

Things that changed during development:

- The whole movement system needed to be remade to eliminate many different conditions and work as a class not functions
- First graphical interface was completely created in gui file, but then I switched to designer to make base of GUI