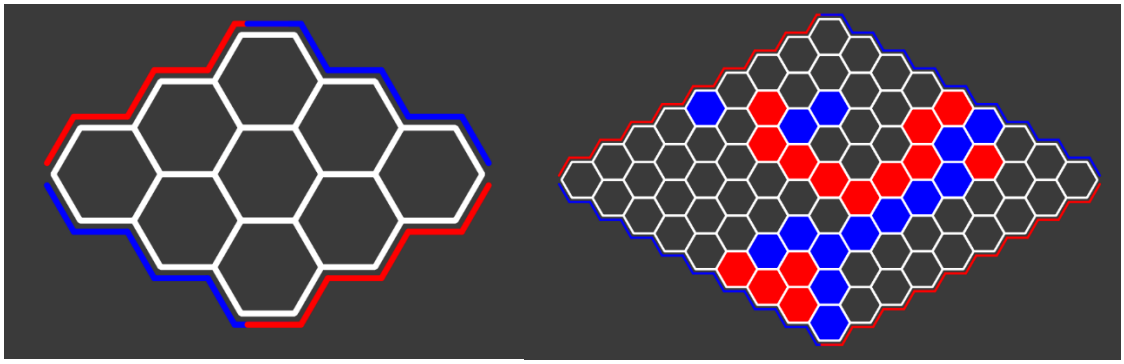
HEX

Main aim and description of the problem

The aim of this project was to create an implementation of the Hex board game using Python. The implementation was based on the architecture shown in the example here. It was also partially based on the example game files for the dots and boxes game, also at the link above. I set out to make a simple, good-looking GUI for the game for two human players to play on a single computer, allowing for the setup of board size and player names and for the game to be played multiple times with various different setups.

Hex is a two-player game played on a rhomboidal game board with hexes as tiles. The aim of each player is to connect two opposite walls with their tiles.



This is an example small version of an empty board and an example of a winning blue connection on a larger board. The top-left and bottom-right walls are outlined with color red while the top-right and bottom-left are outlined blue. This color scheme shows which player needs to connect which walls to win the game – one player places red tiles, the other one places blue ones. In the case of this project, the board is displayed on the screen and allows the players to click the tiles with a single mouse click to make their move when it is their turn. The app then reacts if a pair of walls was connected by the right player and determines the winner. For further information, you can visit the Wikipedia page of the game.

The program's structure

The game's underlying logic runs through three primary classes – HexState, HexGame and HexPlayer.

HexState is the most fundamental class and it contains all the major information about the current state of the game board, who is playing, etc., and defines how a move is made. A HexState object returns a new HexState object once a move is made. Though the state object, important information, such as neighboring tiles, game winner, etc. can be extracted. The object is used only as an attribute of a HexGame object throughout the project and it is envisioned to be this way.

HexGame acts as an interface for changing states, one operates the game through acting on a HexGame object, not through acting on the state itself, so a HexGame object contains two permanently set players and a state which is replaced by a new one every time a move is made. It allows for control of input and encapsulation of the game's state. It also defines the players as the game commences, based on values with which the object has been constructed.

HexPlayer is a more simple class, however it stores data crucial to the progression of the game. It defines the player's name and a unique character used throughout the code to determine hex tile ownership. It also contains given player's connected tiles. The game operates by updating the player's connection sets every time it makes a

move, so that it does not analyze the whole board for a possible winning tile connection every single time a move is made.

Classes added to run the GUI using PySide2 include:

- **HexagonItem** – this class, inheriting from a QGraphicsPolygonItem, represents a single tile in the grid of hex tiles. It defines the shape and look of the hexagon and is created with the aim to make it simple to create a whole grid of such items by adding them to a provided scene instantly when constructed. Also allows to set uniform event effects (e.g. hover and click). The constructor also ties each HexagonItem to its desired coordinates in the game board representation from HexState.
- **InteractiveGraphicsScene** – created to allow for the scene items to emit an 'itemClicked' signal so that hex tiles can be seamlessly clicked and identified. (inherits from QGraphicsScene)
- **ErrorDialog** – defines a pop-up dialog used for error messaging. (inherits from QDialog)
- **HexGridView** – allows for scaling of the graphics view through the use of mouse scroll. (inherits from QGraphicsView)
- **HexWindow** – main GUI class through which the whole game window runs. It defines the setup screen, the game initialization process, the hex grid painting and all the visual aspects of transforming the HexGame object information into an interactive, visible game board. (inherits from QMainWindow)
- **Ui_mainWindow, Ui_errorDialog** – generated through Qt-Designer to define what widgets the main window and the error dialog contain.

User manual

To run the game, an installation of Python and PySide2 is required. You can install the requirements by running the following commands in the terminal (on Linux):

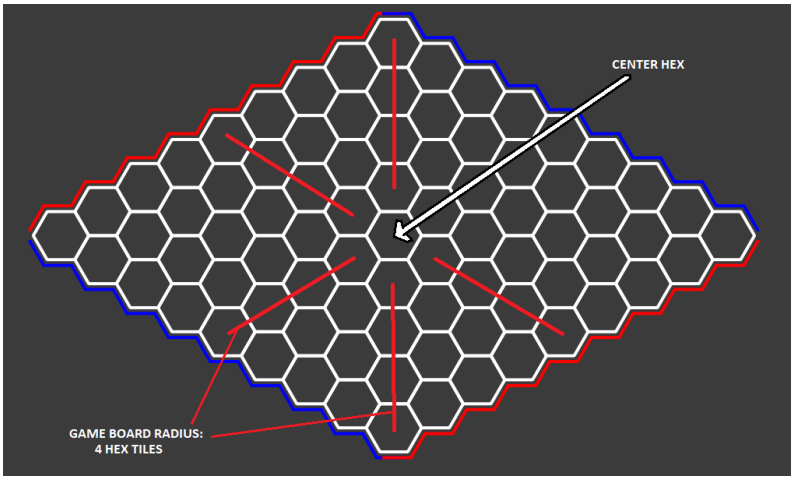
```
sudo apt install python3

sudo apt install pip

pip install pyside2
```

Once you've installed the required packages, you can run the game by opening a terminal window in the game's folder and running the command:

```
python3 main.py
```

This will open the app and present you with the setup window.

Here, you can type desired player names or leave them as Player 1 and Player 2. You can also choose the board radius by clicking the arrows beside the number. The color on the right side of the player name is the color that will represent the player's tiles. Board radius determines board size and is the number of hexes between the center of the board and the sides, as in the example below:



Once you've input all the information, you can click 'START GAME' and be transported to the game screen, on which, at the top, there will be text showing which player's move it currently is and which color this player possesses. Then the player can simply click on the tile he wants to set his marker on. The players then take turns until the winner is determined. You can also restart the game at any time by clicking the 'RESTART' button or exit the game by clicking the 'EXIT' button. Once the game is finished, a prompt will be shown to inform you of this fact and show the name of the winner. Then, you can restart the game or choose to rematch by clicking the 'REMATCH' button which will restart the game with the previous setup, only the player names will be reversed so that the player that has previously moved first will now move second.

Reflection on the project

A lot of time was consumed by making sure that the hex tiles and the outlines are drawn properly. This is not necessarily a bad thing, since I think that, in the end, the game board looks very good and is easy to use and play on. This, however, made it so that some quality-of-life improvements that might have been added to the game, weren't. These might have included things such as a short explanation of the rules of the game that can be shown in the setup screen but I came up with such improvement ideas very late in the making of the project because my focus lay mainly with the game screen's look and functioning. The backend classes, HexGame, HexState and HexPlayer proved to be very reliable and required very little change after initial creation and testing through a makeshift text interface. They held up very well throughout the creation of the GUI, so for the purpose of this implementation, I think they are quite well designed. Some problems with expandability may include the reliance on HexPlayer objects for running the game and making moves which might pose problems with, for example, loading a game from a state if such a feature was to be implemented, but with a continuous game running though a HexGame object, this approach works well. In terms of other game features, perhaps more of them could have been included, such as earlier mentioned instruction manual or perhaps an option to play against the computer or some sort of variation of the rules, unfortunately some of the ideas that I came up with, I came up with towards the end of the project and there was very little time left to accommodate such features. This is why perhaps the game experience may feel too simple at times but overall I think the final state of the project accomplishes what it aimed to accomplish, which is to allow players to simply play the game on a well-drawn game board, with simple controls and info dialogs helping them navigate through it. The late addition of some non-essential features may have made for a more clumsy experience and compromised other areas. All in all, as what it was set out to be, which is an implementation of the game of Hex, it looks good, works well and there weren't many problems with using it or using the code written to further the project.