APT A2 Report

Extension by Guy Witherow (s3783428)

Group Component

**Note: This report was written during the individual component, as it was not completed during the group project duration.*

Use of Linked Lists:

The linked list was used to represent the Bag object, which was used for the Bag object, which contained all the tiles in the game. This was chosen as it was the easiest to shuffle, since the links between each data piece is very easily manipulated and visible.

Use of the STL vector:

The vector was used several times throughout the implementation, but the primary use was for the buffer, which used them to represent multiple variable length lists of tiles waiting to go into the board. This was because it was easy to implement the buffer when you did not have to pre-specify a size, allowing for dynamic vector addressing and usage.

Use of the Array:

The array was chosen to represent the board and the factories. This was because it was known how many we needed, and how long they would be, making it easy to pre-allocates the sizes. For the board it also makes reading code addressing the board a bit easier to read, as it follows a normal co-ordinate system, even with x and y flipped.

Use of ADTs:

The system was split into logical parts, representing parts of the game. This was the most readable way to represent the game, as many functions then take “Factories” or “Tiles.” At times, this does fail, as for example the table is represented by a factory. The inheritance also makes more literal sense, as for example, the bag has a set of tiles inside it. While this may not lead to a perfectly split system in a data availability sense, it massively increases readability.

Efficiency:

The program is reasonably efficient, with no recursive functions or extremely large looping functions. The LinkedList implementation means we have a very quick and easy way to generate tiles, and using a linked list for the bag means that we have a O(1) function to get a tile. This is because it never has to reassign any of the data and can change the “final tile” with a single variable change.

Tests:

This team contributed no tests. This was due to a lack of communication and team participation.

Team Co-ordination and Management:

The team’s time organisation was extremely poor, and there were multiple occasions where people did not deliver their tasks in a timely manner, if at all. The two main issues were that we started too late, and that some people stopped communicating due to, presumably, personal issues or personal time management failures.

Individual Component

Individual enhancement attempted:

3-4 Player mode, and 2 central tables, as well as fixes of the code (not major) and some engine overhauls

Basic overview of corrections made to the original code:

- Game terminates on an EOF
- More comments
- Lid has been implemented and used as a middle ground for tiles at the end of each round
- Random seeds are now actively used
- Command explanations and prompts

For the > 2 player mode, the main issue was overhauling the game logic to look for more than 2 players, as well as initializing the factories array with the correct number. As mentioned above, one of the uses for the arrays in the program, was for things we knew the size of, so having a dynamic number of factories meant making small changes to how pointers were handled.

I made the display show each player side by side, so that players can make informed moves. It already displayed the names of each player above their board, but now displays them side by side.

The save files also needed to be overhauled, but much of that was simple code looping.

I also added a GameEngine object, as passing each separately was cumbersome, and also it allowed me to remove a lot of bulk from the Main file.

There is also an IOClass, which takes a number of the printing and user input functions and keeps them nice and tidy.