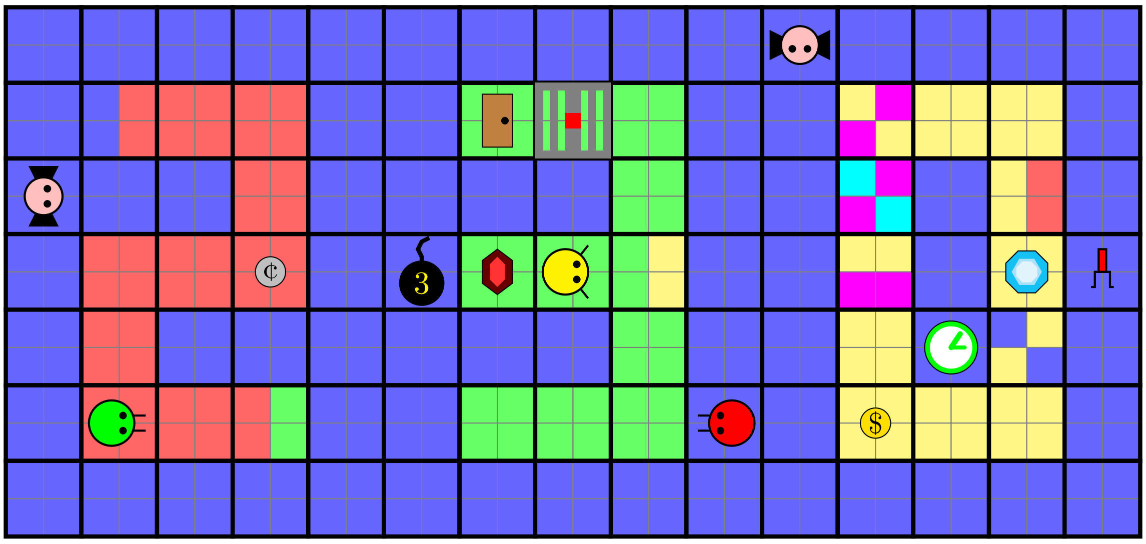
CS-230 Software Engineering
Functional Specification (2022/2023)

Liam O'Reilly
Stewart Powell



1 Introduction

You have been tasked with creating a digital version of the 1997 *Jewel Chase* game which was part of the *Microsoft Entertainment Pack: The Puzzle Collection.* This is a game of strategy, quick response ... and of course a little luck!

There are various YouTube videos that show gameplay of the original game. Some of these are:

- `https://youtu.be/hy9X3ru9YeI`
- `https://youtu.be/VKASKCjRpIc`

The game's rules are detailed in this document. They are not too complex, but not too simple either. Various design decisions have been taken to keep the complexity at a reasonable level and therefore may deviate from the original game. The rules specified here are a little loose. If they specify something then it must be followed, but otherwise you have creative freedom.

It is up to you to design the classes, algorithms and GUI involved in the development of this game.

The gameplay specified in this document **mostly** aligns with the original game, but does differ in places and functionality.

2 Game Title and Theme

You can come up with the title and theme of the game. You may stick with the theme of jewel thieves or you may choose your own by putting some thought into this and making something unique and exciting!

This document describes the type of tiles, NPCs and items used in the game. You may draw them to align them with your theme. For example, you could create a game around robots, dinosaurs, or whatever you like. The items (see Section 3.3) can also be substituted according to your theme as long as they behave as described in this document.

The graphics used in this document are intentionally simple and plain (much plainer than the original game). You can and should produce nicer graphics if you are able to. Even simple image files, when tiled, can look very nice.

3 Overall idea, Components, and Gameplay

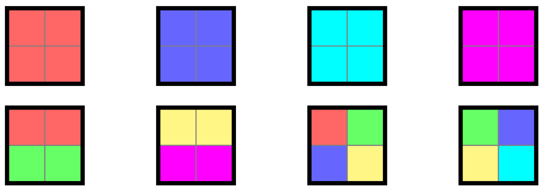
The game is played on a level made up of square tiles that form a rectangular area which the player and NPCs travel within. Your job as the player is to collect as many items as possible – before the other thieves do –

and make it to an exit.

Many of the details below leave certain quantities and durations open. The level files will specify these values (see Section 5).

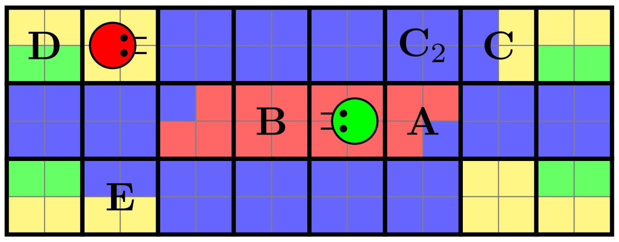
3.1 Tiles and Movement

The tiles that make up the level are all square and the same size. They form a fundamental element of the gameplay with each having four colours (Red, Green, Blue, Yellow, Cyan and Magenta). Some examples of tiles are:



The tiles form a rectangular level. The allowed movement of the player and (some) NPCs are determined by the coloured sections of each tile. When a movement request occurs – for example, let's say right – the player will move to the first tile to their right which has at least one colour in common with their current tile. If no such tile exists no movement occurs.

As an example, let's consider the following level, with the player in green and an NPC in red (note the letters are only here to identify the locations and are not part of the game).



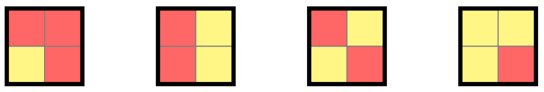
Here is it possible for:

- The player to travel **right** and move to **A**, as it is the first tile with a common colour (red).
- The player to travel **left** and move to **B** as it is the first tile with a common colour (red).
- The NPC to travel **right** and move to **C** as it is the first tile with a common colour (yellow). Note: if the NPC then moved **left** they would end up on $\mathbf{C_2}$.
- The NPC to travel **left** and move to **D** as it is the first tile with a common colour (yellow).
- The NPC to travel **down** and move to **E** as it is the first tile with a common colour (yellow).

The player **cannot** move **up** nor **down** as no tile has a common colour. The NPC **cannot** move **up** as they are at the edge of the level.

Within this level, it is possible for both the player and the NPC to move around the level and reach any tile.

There are 1,296 possible combinations with the order of colours **not** affecting gameplay. For example, the following tiles are all gameplay-equivalent.



The order does however effect the look-and-feel of the level so you might want to be mindful of this when developing levels!

3.2 NPCs

Within the game there are three types of NPCs. The NPCs are:

- Flying Assassins

  The Flying Assassin is a flying NPC which is actually rather dumb. It moves in a straight line in the direction it is facing until it reaches the edge of the level. On reaching the edge, it rotates 180 degrees and continues moving forward. As this NPC is flying, it does **not** respect the floor movement rules and simply always travels straight. The movement is either vertical or horizontal.

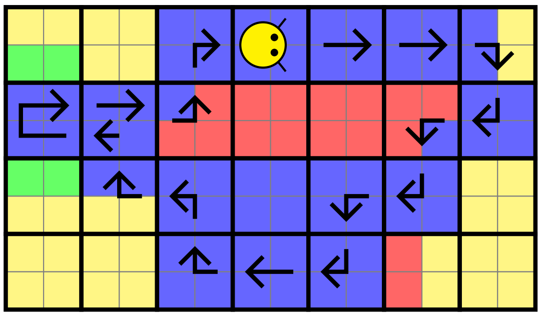
  If the Flying Assassin connects with (i.e. occupies the same tile) the player, the player loses. If they connect with another NPC, that NPC is removed from the game.

  The Flying Assassin is not considered a thief in this document i.e. they do not interact with items in the same way (see Section 3.3) or cause the player to lose (see Section 3.4).

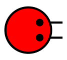
- Floor Following Thieves

  The Floor Following Thief follows the coloured edge of the floor. Each thief is assigned a colour which they must follow (and start on, see Section 5 for more information). Starting in the direction they are facing, the thief will follow the **left-hand** edge of the floor (as if their left hand is touching a wall) and they will follow this around corners.

  As an example consider the following level, with a Floor Following Thief set to the colour blue and starting facing right:



- Smart Thieves



The Smart Thief is the smartest of all the NPCs and their mission within the game is to collect as much loot as possible and reach the exit of the level before the player. They move rather slowly (compared to the player), following the movement rules as set out in Section 3.1, and move as follows:

1. They take the valid move in the direction (up, down, left, or right) that is on the shortest path toward the nearest reachable loot or lever.
2. If there is loot or levers on the level but they are all unreachable then the thief moves in a random but valid direction.
3. Once all loot and levers have been collected, the thief moves in the direction (up, down, left, or right) that is on the shortest path toward the nearest reachable exit. If all exits are unreachable then the thief moves in a random but valid direction.

Note: the shortest paths are calculated following the movement rules as set out in Section 3.1. A smart thief **cannot** move through other NPCs, the player nor bombs.

3.3 Items

There are different items located throughout each level. Each tile can have a maximum of 1 item upon on. When the player or an NPC moves onto a tile that has a collectable item upon it, the item is collected.

The items are:

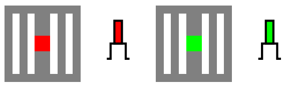
- Loot (¢, $, Rubies, and Diamonds)

Coins, rubies and diamonds are collected by the player to increase their score. The cent coin is worth the least, next the dollar coin, then the ruby, and finally the diamond. Thieves also collect loot which prevents the player from doing so (and has no effect on the player's current score).

- Clocks

Clock are collected by the player and provide additional time for the player to complete the level (see Section 3.4). Thieves also collect clocks which prevents the player from doing so and reduces the remaining time for the player to complete the level).

- Gates and Levers



The player and thieves cannot move onto a tile that has a gate. Levers can be collected by the player and thieves. Upon the collection of a lever, all gates matching the colour of the lever disappear.

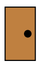
- Bombs

Bombs are one of the most dangerous items within the game, however they are harmless until triggered. A bomb is triggered when the player or any thief occupies a neighbouring tile. Once triggered, the bomb counts down from 3 and then explodes. Neither the player nor thieves can stand on top of bombs.

On explosion, the blast travels the full extent of the level in both the horizontal and vertical directions, destroying all loot, clocks, levers in their path. If the blast hits another bomb, it causes that bomb to explode immediately i.e. a chain reaction. Bomb explosions have **no** effect on gates nor doors.

Note: it is possible to make a level unwinnable by destroying levers before they are collected!

- Doors



Doors represent the exit(s) of a level and cannot be collected. For more information see Section 3.4.

3.4 Winning and Losing the Level

In order to win the level, the player must reach a door **after collecting all loot and levers** before the time expires. You get additional bonus points for the time remaining upon completing the level.

If a thief (i.e. not a Flying Assassin) stands on a door – after all loot and levers are collected, or the time expires – then the player looses the level.

If the player or a thief stands on a door while there is loot or levers still in the level, then nothing happens.

4 Simplifications from the Original Game

The gameplay specified in this document mostly aligns to the original game, but does differ in places and functionality. Here is a non-exhaustive list of some of the more major aspects:

- While the game should be designed and implemented to have good looking graphics, it does not need to have nice/fancy/smooth animations.
- Explosions from bombs need not have nice animations - they may have no animations at all.
- Scrolling the board is not necessary nor advised (you can create the levels so that they fit in the window).

5 Levels

The game will comprise of multiple levels. Upon completing a level the next level will be unlocked. Your player profile will keep track of the maximum level you have unlocked (see Section 6). You can replay all unlocked levels.

Each level is stored in its own file. It is up to you how you design and structure these files. However, they must be simple ASCII based files.

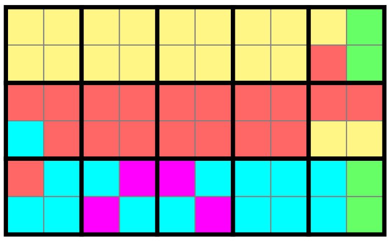
Each level must store the following information (not necessarily in this particular order):

- Size of the level (width and height) - the level need not be square.
- The colours of each tile.
- The starting locations of all NPCs and related data e.g. initial directions, colour to follow for Floor Following Thieves.
- The number of seconds that the player has to complete the level in (see Section 3.4).

A starting point of your design might be as follows:

```
5 3
YYYY YYYY YYYY YYYY YGRG
RRCR RRRR RRRR RRRR RRYY
RCCC CMMC MCCM CCCC CGCG
```

The above specifies the tiles of the level shown below. The first two numbers specify the width and height of the level (which will make parsing the file easier). Following this, each row of the file represents one row of the level. With each tile being separated by a space; each tile is specified by a quadruple of colours.



The rest of the data such as the time to complete the level, items, the NPCs and their initial data as well as the starting position of the player, etc. . .

There are many ways to design this.

6 Player Profiles

The application should support player profiles. Player profiles are used to record game stats for different players. Player profiles can be created and deleted via appropriate menu items or buttons within the application.

Each player profile:

- Has a player name.
- Keeps track of the maximum level that has been unlocked.

7 High Score Tables

The application keeps track of a high score table per level and also allows these to be displayed (per level). Each time a player completes a level, their profile name is recorded on the high score table for that level along with their score. Only the top 10 scores are kept (if a player scores less than all the top 10 for that level then they do not get added to the high score table).

8 Data Persistence

The player profile data is persisted across runs of the application. That is if the user quits the application, then upon reopening the application, the data is not lost.

9 Save/Load Games in Progress

The user shall have the ability to save an ongoing game. This will save the current game state to a file. The user can later load the saved game and resume play. All game data shall be in the same state as they were when the game was saved.

10 Message of the Day

A special Message of the Day (which might change much more frequently than once a day) should be displayed in the app. This message must be displayed in full without modifying any of the characters.

You will need to use a special API (developed for CS-230) to retrieve the message and display it. You should retrieve this message each time the game is launched.

The URL for this service is: `http://cswebcat.swan.ac.uk`

Note: This will involve own research on how to issue HTTP requests, handle HTTP responses, and read the documentation of the Message of the Day API (found by visiting the above URL).

11 Extra Features

This section is not part of A1, but it will be part of A2. It will be helpful for you to plan for extensibility and hence know about this section.

To achieve top marks in A2, you will need to be creative. At a minimum, all the functionality of the Functional Specification should be completed to a high standard. All features should adhere strictly to the specification. You need to get all this working well in order to get a low First Class mark (in A2). In order to get higher marks, you are required to extend the implementation in novel ways. All extensions that do not violate the specification will be considered. Substantial extensions to the software, extra reading and learning, will be required to achieve a high First Class mark (in A2).

12 Forbidden Features

There are forbidden features which you must not design nor implement. These features are reserved for the assessment of CS-235.

Do not design nor implement any of the following features for CS-230:

- A level editor that allows users to create and design their own custom levels via a nice editor.
- Allowing multiple difficulties per level (where for example, the same level at a higher difficulty would have less time, etc.).

13 Libraries and Frameworks

You must program this game in Java using JavaFX. It is strongly recommended using JavaFX's Canvas class to draw the game.

You may use any classes and packages that are part of the standard Java SDK.

You may not use any other libraries or frameworks without first seeking approval. Please use the forums to ask such questions.

Game frameworks will **not** be allowed.