Json files explanation

Vlad Alexandru Robu, Riccardo Speroni
Letizia Maria Chiara Torre, Davide Vola

a.a. 2022/2023

Indice

1 Common Goal Cards

1.1 CG_SHAPE

Cards of type "Shape" are cards which require the player to have a particular shape inside their shelf.

It is possible to decide:

- How many occurrences of the shape that satisfy the conditions are needed to win the card;
- Whether all the occurrences must have the same tile type or if they can differ from each other;
- Whether one eligible occurrence needs all the surrounding tiles to be a different tile type or not;
- The shape itself through coordinates, which identify the shape when drawn on a plane. It is also possible to add other shapes to the same card (such as the mirrored/rotated version).

1.1.1 id

Card IDs are used to identify the card. When adding a new card, it's important to remember to set this parameter to the last card's id + 1.

NOTE: all id values must be in range [0,X], where X is

$$\#numberofcards - 1. \tag{1.1}$$

1.1.2 type

Card type is used to identify which type of card it is.

textbfE.g. if it's meant to be a CG_SHAPE card then its type must be set to "Shape".

1.1.3 numOfOccurrences

Its value must be >= 1. It specifies the number of occurrences of the same shape that are needed to satisfy the card conditions.

textbfE.g. "Two groups each containing 4 tiles of the same type in a 2x2 square". This card requires numOfOccurences = 2, as the card asks for TWO 2x2 squares.

1.1.4 diffType

It has only two valid values (0 and 1):

- When set to 0, all shape's occurrences must be of the same tile type. There is no restriction on which tile type, they all just need to be the same one;

  E.g. "Eight tiles of the same type."

- When set to 1, each of the shape's occurrences can be of a different tile type from the others. They can all share the same tile type or they can all differ (multiple shapes can still share the same tile type);

  E.g."The tiles of one square can be different from those of the other square."

- If numOfOccurrences = 1, then diffType acts as a "don't care" condition, and can be either 0 or 1 without influencing the card.

1.1.5 surrounded

It has three valid values (0, 1):

- When set to 0, multiple neighbor tiles to the shape occurrence can share the same tile type, and they will be considered part of the shape;

  E.g. A 2x2 square is considered a square only if it's a group made of only 4 tiles of the same tile type positioned to make a 2x2 square. if surrounded = 0, then even a 2x2 square with an adjacent tile of the same tile type of the 2x2 square will count as a square, even if the group of tiles is made of 5 tiles instead of 4.

- When set to 1, all adjacent neighbor tiles to the shape occurrence must have a different tile type from it;

  E.g. A 2x2 square will count as a square only if it's a group made of only 4 tiles of the same type positioned to make a 2x2 square. All adjacent tiles must have a different tile type from the square.

1.1.6 stairs

Its value is 0 by default.

- If set to 1 it identifies the 12th CGC on the rulebook ("Five columns of increasing or decreasing height. Starting from the first column on the left or on the right, each next column must be made of exactly one more tile. Tiles can be of any type.");
- If stairs = 1 then surrounded = 0 and numOfOccurrences = 1;
- Must be set to 0 for all cards except for the 12th CGC on the rulebook.

1.1.7 description

It describes the card.

1.1.8 coordinates

This set of coordinates identifies the shape itself that the card requires.
A card can have in its requirements multiple shapes or even the same shape rotated or mirrored; to add these requirements you need to add another set of positions inside the coordinates' node.

E.g. A 2x2 square coordinates will be (0,0) (0,1) (1,0) (1,1) as those coordinates when drawn on a plane identify a square.

E.g. "Five tiles of the same type forming a diagonal." This card requires a diagonal which can be from left to right or from right to left, so we add both of them to the coordinates node:

```
"coordinates":
  [
      [
         {"x": 0, "y": 0},
         {"x": 1, "y": 1},
         {"x": 2, "y": 2},
         {"x": 3, "y": 3},
         {"x": 4, "y": 4}
      ],
  [
         {"x": 0, "y": 0},
         {"x": -1, "y": 1},
         {"x": -2, "y": 2},
         {"x": -3, "y": 3},
         {"x": -4, "y": 4}
      ]
  ]
```

The only requirement is that all sets of positions must have the (0,0) coordinate inside; this means that (0,0) must be part of the shape and because of that sometimes negative coordinates are needed to describe the shape. This doesn't affect the code.

1.2 CG_GROUPS

Cards of type "Groups" are cards which require the player to have groups of tiles without a particular shape inside their shelf. A group is composed of adjacent tiles of the same tile type.

It is possible to decide:

- How many groups that satisfy the condition are needed to win the card;
- The least number of tiles needed to identify an eligible group.

1.2.1 id

The card ID identifies the card. When adding a new card, it's important to remember to set this parameter to the last card's id + 1.

NOTE: All id values must be in range [0,X], where X is

$$\#numberofcards - 1. \tag{1.2}$$

1.2.2 type

Card type identifies which type of card it is. If it's meant to be a CG_GROUPS card then its type must be set to “Groups”.

1.2.3 numOfOccurrences

Its value must be $>= 1$. It specifies the number of groups that are needed to satisfy the card conditions.

E.g. "Six groups each containing at least 2 tiles of the same type”. This card requires numOfOccurences $= 6$, as the card asks for 6 groups.

1.2.4 atLeast

It specifies the least number of tiles needed to identify an eligible group.

E.g. "Six groups each containing AT LEAST 2 tiles of the same type”. This card requires atLeast $= 2$, as the card asks for each group to have at least 2 tiles of the same type. This means that a single tile of a different type from all its adjacent tiles will not count.

1.2.5 description

It describes the card.

1.3 CG_ROWCOL

Cards of type “RowCol” are cards which require the player to satisfy particular requirements regarding the rows and/or columns of his shelf.

It is possible to decide:

- How many rows and/or columns which satisfy the conditions are needed to win the card;
- Whether the card requires only rows, only columns or both.

1.3.1 id

Card ID identifies the card. When adding a new card, it's important to remember to set this parameter to the last card's id $+ 1$.

NOTE: All id values must be in range [0,X], where X is

$$\#numberofcards - 1. \tag{1.3}$$

1.3.2 type

Card type identifies which type of card it is. If it's meant to be a CG_ROWCOL card then its type must be set to "RowCol".

1.3.3 numOfOccurrences

Its value must be >= 1. It specifies the number of rows or columns that are needed to satisfy the card conditions.

E.g. "Two columns each formed by…". This card requires numOfOccurences = 2, as the card asks for 2 columns that satisfy the requirement.

1.3.4 diffUpTo

Its value must be >= 0. Given a row or a column it specifies the maximum number of different tile types inside it. **E.g.** When diffUpTo is equal to 0 it means that the card requires the column to have one of each tile type. Since the game has 6 tile types then the column must have 6 different tile types.

NOTE: diffUpTo = 0 is different from diffUpTo = 6, as 0 means one of each tile type (no repetitions) while as 6 it can have up to 6 different tile types which means some tile types can be repeated.

1.3.5 horizontal

When set to 1, it means the card asks for rows.

E.g. "Two lines each formed by…". This card requires horizontal = 1, as the card asks for rows.

1.3.6 vertical

When set to 1 it means the card asks for columns.

E.g. "Two columns each formed by…". This card requires vertical = 1, as the card asks for columns.

It is possible to have both vertical and horizontal set to 1; in this case, both rows and columns will count towards the card requirements.

1.3.7 description

It describes the card.

2 Living Room

Inside the LivingRoom.json file there's a single node called LivingRoomBoardJSON which contains a 9x9 matrix that represents the board used in game.

Every line follows the format [x,x,x,x,x,x,x,x,x] where every x represents a cell of the matrix.

The number of lines decides the number of rows and the number of x's the columns.

The value assigned to x decides the behavior of the space inside the game, but the final state of the board is ultimately dependent on the number of players currently playing.

- 0 : This space is set to INVALID, no tiles will be placed there;
- 2 : This space is set to PICKABLE if the number of players is >= 2, tiles will be placed there;
- 3 : This space is set to PICKABLE only if the number of players is >= 3, tiles will be placed there. It will be set to INVALID otherwise and no tiles will be placed there;
- 4: This space is set to PICKABLE only if the number of players is >= 4, tiles will be placed there. It will be set to INVALID otherwise and no tiles will be placed there.

3 Personal Goal Cards

Inside PersonalGoals.json there are multiple nodes each representing a singular personal goal card.

The name of the node is the ID of the card in game. All IDs must be in range [1,12].

Each personal goal card must be constructed so that there are exactly six tiles each with a different tile type from the others.

Each card in the file json follows the format ["x", "x", "x", "x", "x", "x"] where the position of the singular “x” represents the row on the personal goal card.

Every row can either have 0,1 or 2 tiles. For every row with 2 tiles there must be an equal amount of rows with no tiles.

Every “x” can either be:

- a line “-” meaning no tiles are required in that row;
- “a:b” This pair defines the position and tile type inside the row: “a” represents a number in range [0,4]: this number is the position inside the row. “b” represents an uppercase letter [C,F,B,G,T,P]: every letter represents the first letter of the tile types used in the game;
- “a:b:c:d” where “c” and “d” follow the same pattern, this means that two tiles are required in that row.