Informe Azul-Game

Datos generales de los estudiantes

- **Lázaro Raúl Iglesias Vera 412** - email
- **Miguel Tenorio Potrony 412** --- email

Introducción

En este informe se describe la implementación del proyecto, así como las estrategias diseñadas para resolver el juego. Además, se definen los diferentes componentes del juego, la simulación del mismo, y otros aspectos de la implementación. La realización del juego se llevó a cabo de acuerdo a las reglas expuestas en el documento, incluidas en el directorio del proyecto, y publicado por los profesores.

---

1. Estructuras de Datos

En esta sección se enuncian las definiciones de los objetos -con sus propiedades- creados para la simulación del juego :

Game:

Representa una mesa de juego, incluyendo los tableros de cada jugador y los expositores de azulejos.

1. *players*: Lista indexada (Lista de la forma: `[<item1>:1, <item2>:2, ..., <itemN>:N]`) de Player, representando a los jugadores del juego.
2. *amount*: Conjunto de Colores representando los azulejos de la bolsa.
3. *outs*: Conjunto de Colores, representando los azulejos que están en la tapa de la caja del juego.
4. *factories*: Lista Indexada de Colores, representando las factorías y sus fichas. Se incluyó el centro de la mesa en esta propiedad por sus características similares con las factorías.

Player:

Representa el tablero, método de juego, y la información de un jugador.

1. *table:* Lista de coordenadas *(X, Y)*, donde una representa una ficha en esa posición de la Pared. De este modo la Pared sería una matriz de 5x5.
2. *score:* El valor numérico de la puntuación del jugador.
3. *strategy:* Nombre del *functor* que representa la estrategia del jugador.
4. *board:* Lista indexada de Line, representando las líneas de patrones.
5. *penalties:* Lista de números, los cuales representan el costo de cada penalización.

Line:

Representa una fila de las líneas de patrones.

1. *stocks:* Lista de colores, donde un color representa una ficha de la fila. Solo contiene fichas de un mismo color o el valor *empty*.

2. *valid:* Lista de colores que se pueden añadir a *stocks*.
3. *all:* Lista de colores que no se han usado en la fila en el transcurso del juego.

Color Set:

Es una lista formada por `Número:Color` donde `Número` es la cantidad de fichas de color `Color` en el conjunto.

Implementación

Para un uso cómodo de estas estructuras, un objeto se define como una lista con valores anotados, donde una anotación representa una propiedad:

```
[<item1>:prop1, <item2>:prop2, ..., <itemN>:propN]
```

Funciones auxiliares fueron creadas para el manejo de estas propiedades, lo cual incluye creación, acceso y eliminación.

Un color representa un azulejo o ficha. Los colores usados son:

- azul
- rojo
- amarillo
- negro
- blanco

Además se usa el valor `empty` para reflejar un espacio vacío donde no hay una ficha, y el valor `first` para representar el marcador de jugador inicial.

---

2. Simulación del juego

A continuación se explica el procedimiento seguido en cada ronda, (según las instrucciones, una ronda está compuesta por las tres fases del juego):

1. Se organiza el orden en el cual van a jugar los jugadores. En la primera ronda, este coincide con los índices numéricos de cada jugador. En el resto, el primer jugador es el que obtuvo el marcador de primer jugador en la ronda pasada, y le siguen el resto de los jugadores ordenados por sus índices. Todo estaría listo para comenzar la **Oferta de Factoría**.

2. ***Oferta de Factoría:*** Se selecciona el jugador al cual le corresponde jugar, este escoge una ficha, actualiza su línea de patrones y su línea del suelo en caso de que sea necesario. De este modo, se va calculando, además, la penalización del jugador, ya que la línea de suelo no puede hacer más que aumentar hasta que esta fase concluya. Al proceso de escoger el azulejo, el cual explicaremos más adelante, le llamamos **estrategia del jugador**.

3. Luego de que le haya tocado su turno a cada jugador, se comprueba que queden fichas en los expositores (esto incluye al centro de mesa). Si no hay más fichas, entonces se procede al **Alicatado de Pared**, en caso contrario, se regresa al paso 2.

4. ***Alicatado de Pared:*** Por cada jugador, se revisan sus líneas de patrones, en caso de haber alguna completa, se coloca la ficha correspondiente en la Pared, y se limpia esa fila, rellenádola de valores `empty`, con las restantes fichas yendo a la tapa o *outs*. Al terminar esta revisión, según los azulejos nuevos, se calcula la puntuación del jugador en esta ronda; esto es, sumarle a la puntución actual -a la cual se le restó la penalización en el punto 2- la puntuación adquirida en la ronda, y se actualizan sus propiedades.

5. Terminada la fase anterior, se chequea si alguna fila de la Pared está completa. En caso afirmativo, el juego ha terminado, por lo que se procede a calcular las puntuaciones de todos los jugadores dado la distribución final de sus Paredes. Luego se termina la simulación no sin antes enunciar las puntuaciones finales. De lo contrario, se prepara la siguiente ronda.

6. La ***preparación de una ronda*** concluye con el paso 1 y se inicia calculando quién será el jugador inicial de la misma, dado el portador del marcador de primer jugador, el cual procede a ocupar el centro de la mesa de forma solitaria. En este mismo instante se penaliza al antiguo portador del marcador con un punto. Luego se revisa si en la bolsa hay suficientes fichas para ocupar todas las factorías; si esto no es posible, se rellena la misma con los azulejos de la tapa. A continuación, se completan las factorías, y se actualizan los valores de la bolsa antes de pasar al paso 1.

---

3. Estrategia del jugador

Fueron ideadas e implementadas 3 estrategias:

1. ***basic:*** De todas las posibles jugadas que permitan colocar fichas en la línea de patrones, se escoge la primera que se detecte. Si esto no es posible, entonces se escoge la primera opción válida de selección de azulejos, y se colocan en la línea de suelo del jugador (o en la tapa si esta está llena).
2. ***greedy:*** Por cada posible selección de fichas que permita colocar un conjunto de estas en la línea de patrones, se escoge la que más mejore la puntuación actual. Para esto se asume que la ronda terminó y se simula un *Alicatado de Pared* para este jugador. El lector puede habilmente detectar que solo es posible obtener una puntuación positiva en caso de que la jugada seleccionada provoque el relleno de una fila de la línea de patrones. De no ser posible jugar en la línea de patrones, entonces se escoge, de las opciones válidas de selección de azulejos, la que menos penalización provoque para el jugador.
3. ***fill column***: Similar a la estrategia *greedy*; la diferencia es la siguiente: De las jugadas que, provoquen mayor crecimiento de las columnas de azulejos de la Pared, escoge la que maximiza la puntuación actual.

---

4. Detalles del Desarrollo

Se desean destacar algunas características del proyecto:

- El proyecto, desde sus inicios, tiene su repositorio en github, con un flujo de trabajo común al empleado en la comunidad de desarrollo. Esto se refleja en las convenciones usadas para las ramas, commits, los issues, pull requests, project, y demás recomendaciones de la comunidad de github.

- Se dispone de un readme y makefile para una fácil interacción con el programa.

- Se implementó un Logger en Prolog para facilidades del desarrollo y la exposición de los pasos, eventos y estructuras de datos del juego desde su comienzo, tratando siempre de enseñar mensajes claros y amigables.
- Se creó una suite de unit tests para las cláusulas usadas.