Introducción a la Programación
Dominio de webvaluación 1
2020s2

Gobstong Us

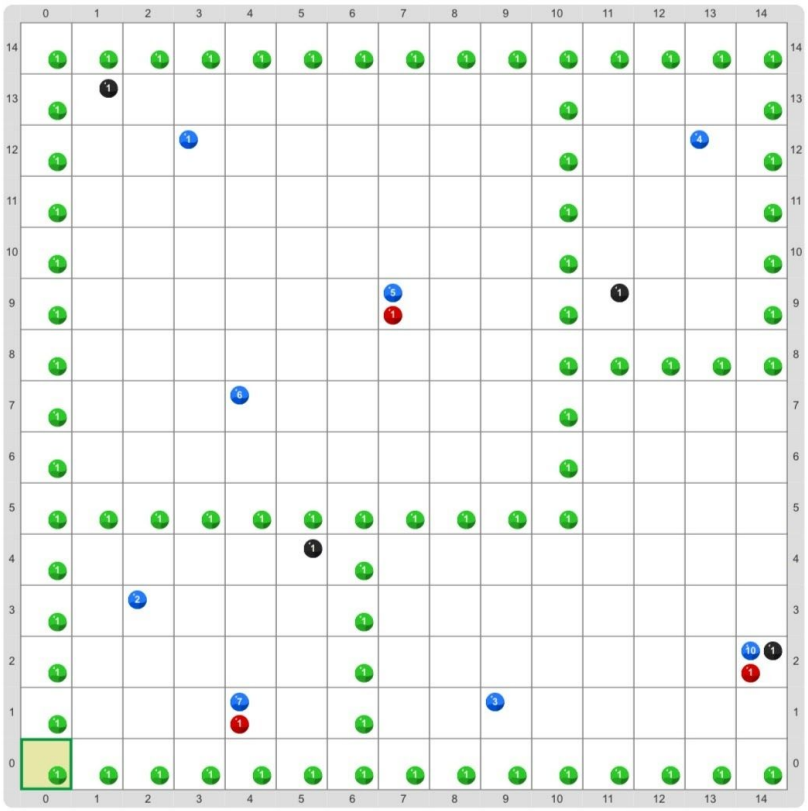
Gobstong Us es un videojuego multijugador en línea en donde los ***Jugadores*** toman el rol de ***Tripulantes*** de una nave que debe ser reparada o de ***Impostores*** que intentan engañar a los tripulantes haciéndose pasar por éstos, pero saboteando la nave y matando a los tripulantes en lugar de repararla. Así, el juego se transforma en una carrera entre tripulantes e impostores, donde ganarán los tripulantes si logran reparar la nave o descubrir quienes son los impostores antes de que los impostores logren matar a todos los tripulantes o sabotear completamente la nave. El juego se desarrolla en una nave que consiste en una serie de ***habitaciones*** y que pueden estar ubicadas de diversas formas según el nivel del cual se trate.
Los impostores, a diferencia de los tripulantes, pueden moverse por las ***rejillas de ventilación***, acortando el camino y permitiéndoles matar a más tripulantes rápidamente. Así, hay pistas que pueden indicar si un jugador cualquiera es un impostor o no. Por ej. si un jugador está sobre una rejilla de ventilación, es probable que sea un impostor, por lo que los otros jugadores tienden a identificarlo y expulsarlo de la nave antes de que mate a nadie o sabotee la nave. También puede pasar que quien se encuentra sobre una rejilla no sea un impostor, y entonces los jugadores lo expulsan injustamente de la nave.

Otra cuestión a tener en cuenta es que los impostores no matan a nadie frente a testigos, para evitar que los identifiquen. Así, sí en una habitación quedan solos dos jugadores y uno de ellos es un tripulante y otro un impostor, el impostor matará indefectiblemente al tripulante (para lo cual debe pararse primero a su lado). Asi, otra forma de identificar personajes sospechosos es cuando estos se pegan entre sí. Dos personajes muy juntos puede entonces ser señal de que uno intenta matar al otro.

En la implementación actual el tablero representa la totalidad de la nave, y una serie de celdas con una bolita **`colorPared()`** marcan las paredes dividiendo la misma en sus habitaciones *(por simplicidad no hay puertas que comuniquen a las habitaciones entre sí en esta implementación)*. Cada habitación posee ***zonas***, dadas por aquellas celdas del tablero que la conforman, En una zona de una habitación puede haber un jugador (como máximo), el cual se identifica con bolitas de **`colorJugador()`** tantas bolitas como el número de identificador de ese jugador (no hay dos jugadores en todo el tablero con un mismo identificador, los identificadores son siempre mayores que cero). Aquellos jugadores que son impostores tienen *además* una bolita de **`colorImpostor()`**, que identifica tal característica; aquellos que no la tienen, son tripulantes (observar que por sí sola una bolita de este color NO alcanza para marcar un impostor; debe ser un complemento a la representación del jugador). Una única

bolita de color **`colorRejilla()`** en una celda indica una rejilla de ventilación. Es importante entender que el tablero representa la totalidad de la información, y no lo que vería cada jugador en particular.

A continuación hay un tablero de ejemplo que muestra algunos de los posibles niveles del juego y algunos jugadores en él (donde **`colorImpostor()`** es **`Rojo`**, **`colorPared()`** es **`Verde`**, **`colorJugador()`** es **`Azul`** y **`colorRejilla()`** es **`Negro`**)



En esta webvaluación lo que nos interesará es obtener información o modelar pequeños aspectos del juego, y no el juego en su totalidad. Para que mirar los elementos de una habitación sea sencillo, se brindan las siguientes primitivas:

```
// Para recorrer una habitación
IrAlPrincipioDeLaHabitaciónAl_Y_(dirPrincipal, dirSecundaria)
hayPróximaZonaEnLaHabitaciónAl_Y_(dirPrincipal, dirSecundaria)
IrAPróximaZonaEnLaHabitaciónAl_Y_(dirPrincipal, dirSecundaria)

// Para recorrer las diferentes habitaciones
IrAHabitaciónNúmero_(númeroDeHabitación)
cantidadDeHabitacionesEnLaNave()
```

Consignas 1 Webevaluacion:

1) Escribir la función hayTripulanteAcá que indica si en la zona actual hay un tripulante (recordar que los impostores son jugadores, pero no tripulantes…).

2) Escribir la función cantidadDeTripulantesEnLaNave que describe la cantidad total de tripulantes que hay en la nave.

3)Escribir la función habráUnaExpulsiónInjusta que indica si los jugadores van a determinar expulsar a alguien de la nave injustamente. Esto ocurre cuando en cualquier lugar de la nave un tripulante está sobre una rejilla de ventilación.

4) Escribir el procedimiento AcercarseParaMatar que hace que el impostor se mueva una posición en forma ortogonal (a una de las zonas lindantes) con la intención de acercarse al impostor que está cerca, para matarlo posteriormente. Para esto se presupone que en la zona actual hay un impostor y que en alguna de las celdas en diagonal a la actual (hacia cualquiera de las 4 diagonales) hay un tripulante, y que estos jugadores son los únicos dos en la habitación. Así, por ej. si el tripulante está en la diagonal Norte-Este, entonces el impostor debe moverse a la celda al Norte o al Este.

5) Se quiere saber si en algún lado hay más de un impostor agazapado esperando una víctima, es decir, si en una habitación hay 2 o más impostores y ningún tripulante. Se pide, entonces implementar la función hayImpostoresAgazapadosEnAlgunaHabitación que indica lo mencionado.