Trabajo Estructuras de Datos 2019-2

Integrantes: Máximo 4

El objetivo del juego será desarrollar una versión modificada de PAC-MAN en Java.

Se adjunta un código base que genera un pac-man y dos fantasmas. El pac-man tiene movimiento con las teclas de dirección del teclado. Analice el código y entienda su funcionamiento para posteriormente hacerle modificaciones y continuar el desarrollo

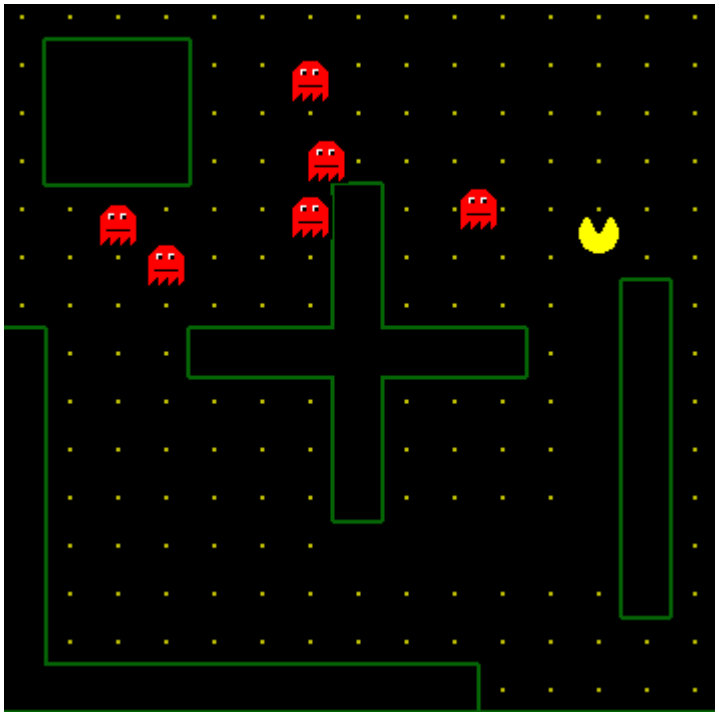
1. **(5%) Multijugador**: Existirán dos jugadores, cada uno tiene una cantidad de pac-man disponibles en la partida. El segundo jugador moverá su pac-man con las teclas WASD. Los posibles colores de pac-man para el jugador 1 son: amarillo, café, rosado. Los colores posibles para el jugador 2 son: azul, rojo y verde. En el proyecto se adjunta una imagen de un pac-man de cada color. Puede utilizar la siguiente herramienta para girar un gif: https://ezgif.com/rotate/

   Nota: Por facilidad, se cambia de imagen cuando el pac-man cambia de dirección, pero también puede girar la imagen desde Java con una transformación de X grados. Si lo quiere hacer así puede consultar en: https://stackoverflow.com/questions/8639567/java-rotating-images

2. **(5%) Fantasmas:** La cantidad inicial de fantasmas se generará de manera aleatoria, dentro de un intervalo solicitado al usuario por consola al inicio del juego (mínimo de fantasmas, máximo de fantasmas). Todos los fantasmas por defecto inician siendo rojos.

3. **(10%) Obstáculos**: Trate el tablero como una cuadrícula, en la cual cada celda tiene un tamaño de 30px. Los obstáculos son lugares por los cuales no se puede mover los pac-man ni los fantasmas. Antes de iniciar el juego, se le debe solicitar al usuario por consola la cantidad de obstáculos que tendrá la partida, y luego para cada obstáculo se le debe solicitar la celda inicial del obstáculo y la celda final, de modo tal que forme un rectángulo. Las celdas inician en 1 en la esquina superior izquierda y aumentan positivamente hacia la derecha y hacia abajo.
   Por ejemplo, para la siguiente imagen se pudo haber ingresado:

   cantidad de obstáculos = 6
   obstáculo 1: desde celda (2, 2) hasta la celda (4, 4)
   obstáculo 2: desde la celda (1, 8) hasta la celda (1, 15)
   obstáculo 3: desde la celda (2, 15) hasta la celda (10, 15)
   obstáculo 4: desde la celda (5, 8) hasta la celda (11, 8)
   obstáculo 5: desde la celda (8, 5) hasta la celda (8, 11)
   obstáculo 6: desde la celda (14, 7) hasta la celda (14, 13)



Por facilidad puede pintar los obstáculos de algún color relleno, puede utilizar el método g.fillRect(x, y, ancho, alto) en el método paintComponent para dibujar un rectángulo. Con el método g.setColor(Color.GREEN); establece el color (Verde para ese ejemplo) para pintar (Se debe llamar antes de establecer dibujar el rectángulo)

Utilice una estructura de datos idónea para almacenar los obstáculos. Cree la clase Obstáculo.

4. **(10%) Posición inicial**: Los pac-man iniciarán en una posición aleatoria diferente cada uno. La posición debe ser en una de las 3 filas inferiores de la pantalla, que no tenga un obstáculo (se supondrá que el usuario no creará un obstáculo de un tamaño tal que obstruya todas las ultimas tres filas). Los fantasmas iniciarán desde dos puntos aleatorios generados en las tres filas superiores que no tengan obstáculo (Se hace igual suposición), de los cuales la mitad parte de cada uno de los puntos.

5. **(25%) Movimiento**: El movimiento de los pac-man y de los fantasmas se ve restringido por los obstáculos, es decir que no se puede transitar por una celda que tenga un obstáculo. Se debe detectar la colisión con los obstáculos.
   Los pac-man siempre se moverán en la dirección que el usuario indicó con el teclado (Como el código entregado) hasta que el usuario cambie de dirección, o se encuentre con un obstáculo. En este último caso el pac-man debe parar y solo reanudará el movimiento cuando el usuario cambie la dirección.
   Los fantasmas se deben mover de manera aleatoria en una dirección hasta que se choquen con un obstáculo, momento en el cual deben elegir la dirección siguiente

de manera aleatoria. La dirección inicial de cada fantasma se debe establecer de manera aleatoria.

6. **(15%) Bolas de poder:** Al iniciar la partida se generará cierta cantidad de bolas de poder, entre un máximo y un mínimo solicitado al usuario por consola al iniciar el juego. Cada bola de poder se ubicará aleatoriamente en una celda que no tenga obstáculo. Cuando un pac-man pasa sobre una bola de poder convierte a todos los fantasmas en fantasmas azules durante 8 segundos, los pacman duplican su velocidad durante dicho tiempo y los fantasmas la disminuyen a la mitad. Utilice la imagen del fantasma azul para mostrarlos de esa forma durante ese tiempo.

7. **(10%) Muerte**: Un pac-man muere cuando un fantasma rojo pasa sobre él. Un fantasma azul muere cuando pac-man pasa sobre él.
   Cada jugador tiene una reserva de 3 pac-man al inicio del juego. El color de cada pac-man es aleatorio, pero desde el inicio del juego se le deben asignar a cada jugador. Al iniciar el juego por consola se deben mostrar el color de cada pac-man asignado a los jugadores, en orden de como irán saliendo a la partida.
   Cuando un pac-man muere, el siguiente en la cola sale inmediatamente desde la posición de inicio del jugador. Por cada 3 fantasmas que un jugador mata, obtiene un nuevo pac-man que pasa de ultimo a la lista de pac-man del jugador. Cuando un jugador obtiene un nuevo pac-man, se debe mostrar por consola el mensaje: Jugador X ganaste un nuevo pac-man, seguido de la lista de pac-man del jugador en orden de como saldrán.
   Utilice una estructura de datos adecuada para representar esta situación.

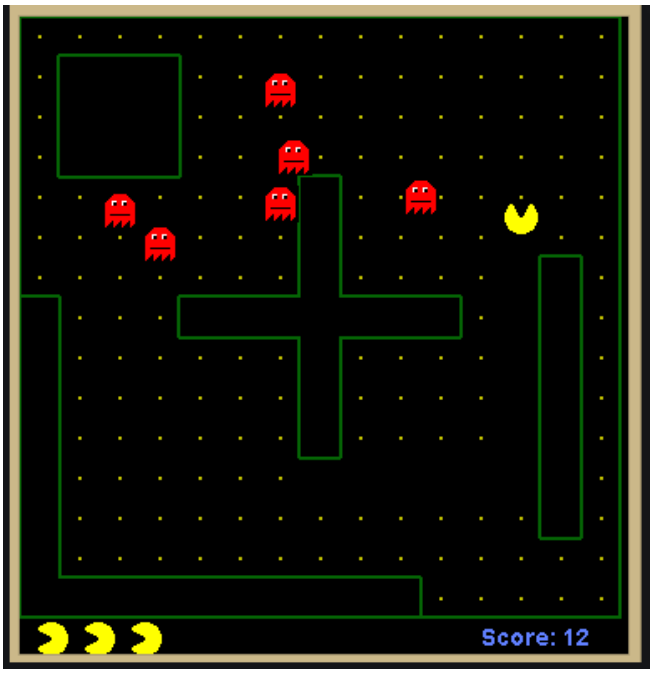
8. **(10%) Comida**: Las celdas donde no hay un obstáculo ni una bola de poder deben ser rellenadas con granitos comida. Utilice el método fillOval(x, y, ancho, alto) para dibujar la comida. Cree la clase GranitoComida.

   Cuando un pac-man pase sobre un granito de comida, se la comerá e irá directo a su estómago :P Pero los pac-man son un poco desagradables y vomitan 5 granitos de comida (o los que tenga, si ha comido menos de 5) por cada fantasma que el pac-man rival mata. Cuando vomita, los granitos de comida caen de manera aleatoria en cualquier celda (Si caen sobre un obstáculo simplemente se desaparecen, al igual que si caen sobre otra comida). Los granitos que un pac-man vomita son los últimos que entraron a su estómago. Utilice la estructura de datos adecuada para representar esta situación.

9. **(10%) Fin de partida**: La partida finaliza cuando todos los granitos de comida del tablero se terminan, o cuando todos los pac-man de un jugador se mueren. En el

primer caso, el ganador de la partida será el que más granitos de comida tenga en su estómago, mientras que en el segundo caso el ganador será el jugador que aún queda vivo. Al finalizar la partida se deberá mostrar en pantalla un letrero que diga quién es el usuario ganador. Utilice el método g.drawString(cadena, x, y) para dibujar una cadena en pantalla.

10. **Visualización de puntajes** (Opcional): Muestre en la parte inferior de la pantalla los pac-man que cada jugador tiene, en orden de como saldrían y con el color de cada uno. También muestre la cantidad de bolitas de comida que tiene cada uno en su estómago, similar a la siguiente imagen, pero teniendo presente las adaptaciones para el juego modificado. Este punto es opcional, al realizarlo tendrás una bonificación de 10% (La nota máxima sigue siendo 5.0 :P)



Nota: El trabajo deberá ser sustentado en la fecha indicada, en el horario de la clase práctica. La asistencia de todos los integrantes del equipo es obligatoria y la única excusa para no asistir es aquella validada por la universidad. Se seleccionará aleatoriamente un integrante de cada equipo para sustentar, así que asegúrate de incluir en el trabajo solo aquellos que realmente trabajaron. Si la sustentación se realiza de manera adecuada, la nota final es la nota del trabajo según la calificación de los parámetros. Si la sustentación no se realiza de manera adecuada, la nota final del trabajo se puede ver disminuida hasta en 1.5