

**Docente**

Marlene Duarte

**Tema**

Estructura de Datos II

**Tema del laboratorio**

Pacman

**Estudiante**

Nicolás Buitrago Suarez

**Programa**

Ing. de Sistemas

**Octubre de 2017**

Tabla de contenido

**Introducción1**

**Descripción general1**

**Requerimientos funcionales1**

**Requerimientos no funcionales2**

**Descripción técnica de la solución2**

**Pantallazos del juego3**

**Conclusiones5**

**Referencias6**

**Pacman con algoritmo de camino mínimo**

**Introducción**

El laboratorio se desarrolló con el fin de evaluar la implementación de un algoritmo de ruta mínima entre dos puntos de un grafo, encontrando la ruta más eficiente, dándole así un camino a recorrer al enemigo del juego de Pacman.

Debido a que el curso gira en torno a la implementación y uso de los grafos en sistemas informáticos, con este laboratorio se evalúa la resolución de problemas de caminos mínimos dentro de un grafo, tomando en cuenta el peso de cada camino para encontrar el más corto. De esta manera el fantasma, enemigo en el juego de Pacman, puede encontrar siempre el camino más corto hacia el jugador que se mueve por el tablero.

**Descripción general**

En el proyecto se le debía dar vida al juego de Pacman, teniendo en cuenta ciertas modificaciones al juego, de las cuales una de las más importantes es que el enemigo en el juego, el fantasma, implementa un algoritmo de camino mínimo para su inteligencia (Dijkstra). De esta manera, el enemigo persigue todo el tiempo al jugador principal y tiene movimiento propio. Para la implementación del Dijkstra se formó un grafo con cada uno de los cuadrantes que conforman el tablero del juego a los que el usuario puede acceder, es decir, los cuadrantes negros. Su adyacencia está dada por los cuadrantes vecinos que tiene en la matriz que almacena cada uno de los cuadrantes del tablero del juego y que también hacen parte del grafo. Por esta razón, la distancia entre cada par de nodos adyacentes es 1. Por último, cabe destacar que el juego debía ser desarrollado en Java.

**Requerimientos funcionales**

* Compatible con el sistema operativo Windows.
* Ejecutable en computadores personales PC.
* Mostrar en pantalla el tablero del juego.
* Moverse sobre el tablero de juego.
* Calcular la ruta mínima entre el pacman y el fantasma y con este darle vida al fantasma.
* Indicar el puntaje ganado hasta el momento por el usuario.
* Ubicar 10 puntos en posiciones específicas alrededor del tablero
* Mostrar la hora actual de sistema y la hora estimada de llegada.
* Sumar al puntaje cierta cantidad después de consumir un punto y un punto por cada segundo transcurrido.

**Requerimientos no funcionales**

* Permitir reiniciar el juego con un botón.
* El programa es capaz de funcionar eficiente.
* El juego fue desarrollado con una estructura y un diseño que le permitiera ser escalable y no limitar de esta forma futuras actualizaciones.

**Descripción técnica de la solución**

El código fuente está organizado en los paquetes Audio, Sprites y pacman. En Audio se encuentran almacenados todos los audios utilizados para la ambientación del juego. Incluyendo el sonido que se reproduce al inicio del juego, un sonido que se reproduce durante todo el tiempo en el juego, el sonido que se reproduce cada vez que el pacman come alguno de los puntos y por último los dos sonidos que dependiendo el resultado del juego se reproduce uno de los dos al final del juego, uno de ellos se reproduce si el pacman y otro si muere.

En el paquete Sprites, se encuentran todas las imágenes usadas para desarrollar el juego. La imagen del título del juego de Pacman y cada uno de los sprites de Pacman y del Fantasma se encuentran allí. Además, en este paquete se encuentra también la imagen del muro con la que se llenó cada uno de los espacios del juego.

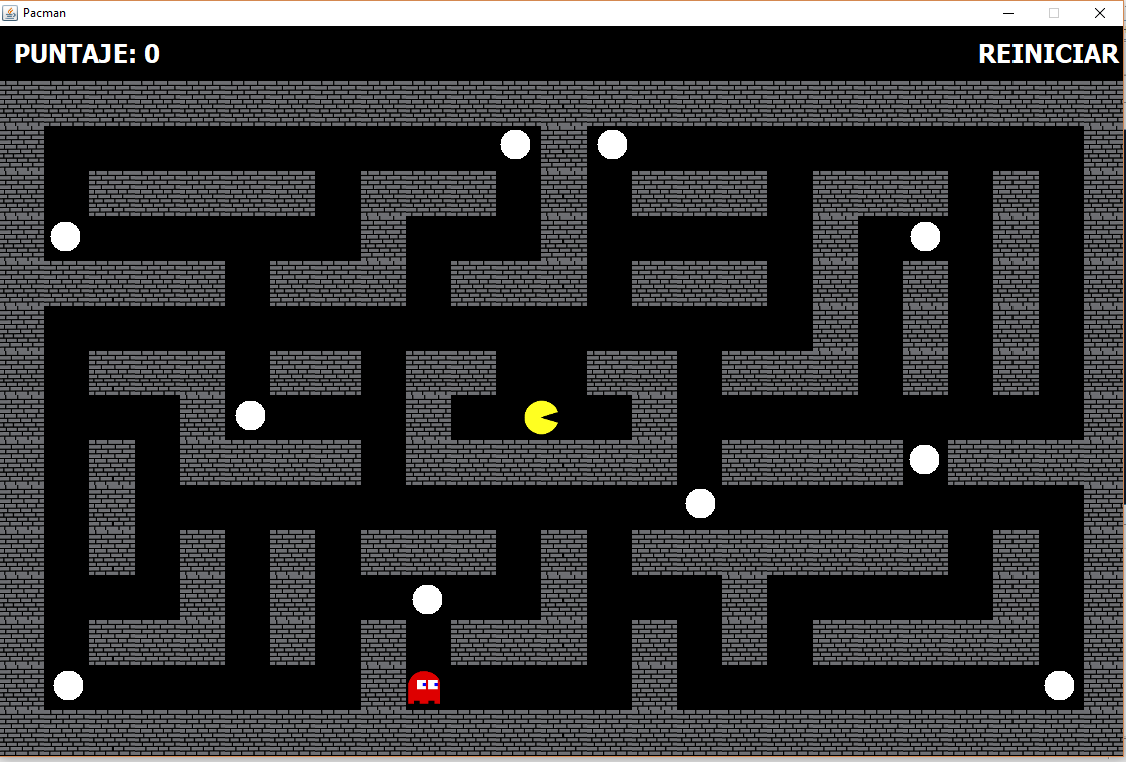
En el paquete Pacman se encuentran cada una de las clases utilizadas para el desarrollo del juego. Entre estas están las dos vistas usadas, la pantalla de inicio donde se le muestra al usuario la opción de salir o de empezar con el juego. La otra vista usada es la principal que es donde se desarrolla todo el juego, esta vista contiene tres JLabel, uno donde se muestra el puntaje del usuario, otro donde se le muestra al jugador si ganó o perdió y el tercer JLabel cumple un papel similar al de un botón cuya función es reiniciar el juego. Por último, en esta misma vista principal se muestra un canvas en el cual se desarrolla el juego. Desde la clase Principal se crean los listeners para el teclado que permiten el movimiento del pacman y el listener del mouse que le permite al JLabel darle la posibilidad al usuario de reiniciar el juego. Además, la clase principal es la encargada de crear el hilo donde se le dará el movimiento al Pacman y desde donde se manejará todo lo relacionado con el dibujo sobre el canvas.

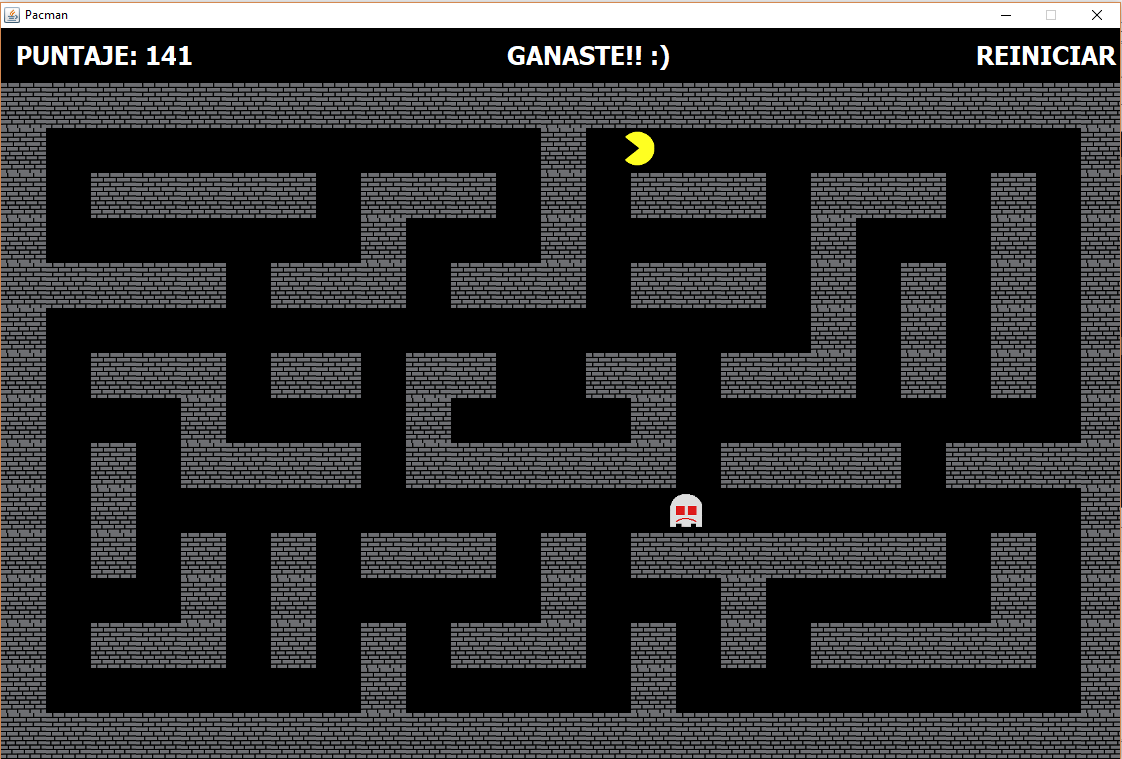
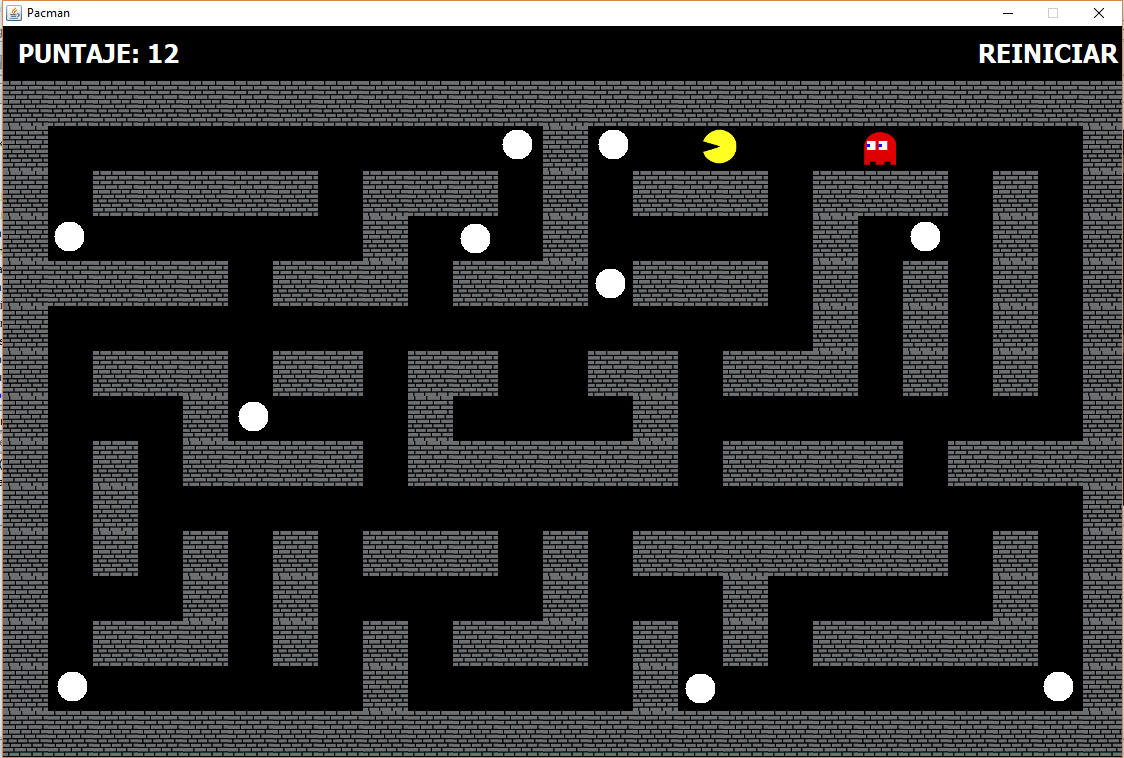
Además, en el paquete pacman se encuentran las clases Tablero, Personaje, Pacman, Fantasma, Sound, Cuadrante y Animation. La clase Tablero, se encarga de todo lo relacionado con el tablero de juego, es decir, todo lo mostrado en el canvas. Este se encarga de crear, almacenar y manejar la información de todo aquello que tenga que ver con el mapa o mudo del juego. Por esta razón es la encargada de manejar los puntos sobre el tablero y de encontrar el camino más corto hacia una posición específica dentro del tablero. Además, esta se encarga de verificar hacia donde le está permitido el movimiento de los personajes usando para determinar esto la técnica de la bounding box. Esta clase posee dos clases privadas una de ellas es Punto que ayuda a manejar la información de cada uno de los puntos que puede alcanzar el Pacman para subir su puntaje y que además ayuda a reubicarlos una vez que el fantasma alcanza alguno de estos puntos. La otra clase privada dentro de esta clase es Camino, la clase que ayuda a implementar el algoritmo de recorrido mínimo usado para darle movimiento al fantasma. El algoritmo de recorrido mínimo implementado sobre el grafo que almacena la clase Tablero es el algoritmo Dijkstra.

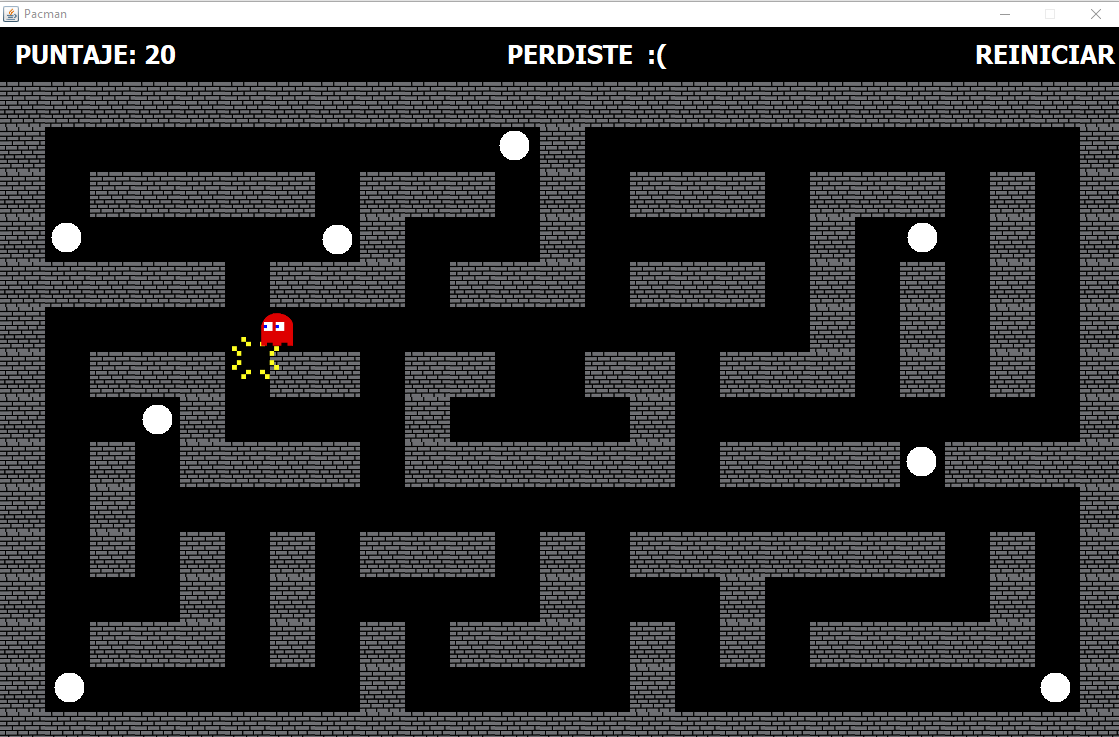
La clase Cuadrante es la unidad en la que se subdivide el tablero del juego. Dada la matriz mundo por el enunciado del laboratorio, cada uno y cero dentro de esta matriz viene representando un Cuadrante en la clase Tablero. Cuadrante almacena la posición de este mismo, si por este se puede caminar, si almacena este algún punto y además almacena un nombre. Se maneja una constante para su tamaño la cual es almacenada en la clase Tablero.

Las clases Personaje, Pacman y Fantasma se encargan de modelar los dos personajes en el juego, siendo Pacman y Fantasma clases hijas de la clase Personaje. La clase Personaje es una clase abstracta que se encarga de modelar lo que hace y lo que conforma a un personaje en el juego, luego ya desde sus respectivas clases hijas se les da forma dependiendo de las características únicas que cada uno de los personajes poseen. La clase Pacman posee una clase privada llamada Puntaje para permitirle un fácil manejo a su puntaje. La clase Fantasma se encarga de crear el hilo que más tarde será utilizado para calcular con Dijkstra el camino mínimo que deberá recorrer para poder alcanzar al pacman.

Por último, está la clase Sound la cual implementa la interfaz Runnable y por lo cual es la que permite desde un hilo aparte el manejo de cada uno de los sonidos usados para la ambientación del juego.

**Pantallazos del juego**

****

****

**Conclusiones**

Los grafos son útiles para representar sistemas de tipo topográfico de forma matemática. Gracias a esto, algoritmos como el desarrollado para este proyecto son útiles para modelar sistemas de la vida real y dar solución a los problemas que estos puedan presentar. De hecho, los algoritmos mínimos se utilizan en diversas clases de juegos como una forma de darle inteligencia a algún enemigo del juego que desea alcanzar al personaje principal, tal y como en el caso del juego que tuvimos que desarrollar. Estos dentro de los juegos también se pueden usar para hacer un manejo efectivo de los recursos dentro de este.

Fue necesario investigar e implementar algoritmos para calcular la distancia entre un punto y una recta, puesto que los puntos de partida y de llegada no necesariamente coinciden con los vértices del grafo. De esta forma se pudo ubicar el punto de partida dado por el usuario sobre la arista más cercana, dado la libertad de colocar dichos puntos en cualquier parte del mapa.

El método detrás de encontrar el camino más cercano fue el uso del algoritmo de Dijkstra, utilizado en la búsqueda de caminos más cortos entre dos vértices de un grafo, similar al problema propuesto para el proyecto. El algoritmo de Dijkstra funciona en grafos conexos direccionados y no direccionados con pesos no negativos en sus aristas. Dicho algoritmo inicializa la distancia las distancias entre los vértices como infinitas, luego evalúa los vértices alrededor del vértice en el que está ubicado para encontrar la ruta más corta. En caso de encontrarlo, se ubica en ese vértice y repite el proceso hasta llegar al punto de partida mientras acumula la distancia del camino que va recorriendo e identificando qué vértices ya ha visitado.

Como se mencionó antes, el algoritmo calcula primeramente la distancia entre un punto dado y la arista más cercana, para calcular desde ahí la ruta más cercana hasta el punto de llegada. Sin embargo, durante el desarrollo del juego se identificaba de forma errónea cuál era el nodo hacia el cual el fantasma se debía dirigir para poder atrapar al pacman. Al realizarse esta operación dentro de una validación, el nodo o Cuadrante que se identificaba como más cercano siempre era el primer cuadrante ubicado en la parte superior del grafo. La solución dada a esta problemática fue validar a partir de la posición en la matriz de cuadrantes en la clase Tablero su adyacencia con los demás cuadrantes y de esta manera después a partir de su posición poder calcular cual era el nodo más cercano.

**Referencias**

* [1] M. Yan, “Dijkstra’s algorithm,” Massachusetts Institute of Technology. regexstr, 2014.
* [2] J. Morris, “Dijkstra’s algorithm,” University of Auckland, 1998.
* [3] M. Deza, *Encyclopedia of Distances,* Nueva York, 2016.
* [4] “Cosmos”, OpenGenus, Available: <https://github.com/OpenGenus/cosmos>