Juego en HTML y JavaScript (Pac-Man)

Los fantasmas del juego “Pacman” creado por Toru Iwatani nos muestra el ejemplo claro de un agente estímulo respuesta, utilizando el algoritmo A\* (para pathfinding); además, la página web gameinternal.com, en su publicación “Understanding Pac-Man Ghost Behavior”, nos menciona que para perseguir al protagonista, en cada uno de los fantasmas se emplea una modalidad distinta, mas no menciona que se utilicen algoritmos distintos entre cada fantasma. Por lo que podemos deducir que cada fantasma ocupa de distinta manera el algoritmo A\*, perteneciente a las búsquedas heurísticas. Teniendo esto en claro, surge mi curiosidad: al desarrollar videojuegos que contengan enemigos capaces de perseguir al personaje en un entorno 2D similar al de Pacman

¿A qué llamamos juegos basados en Pacman? Pacman es un videojuego de una bola amarilla que avanza comiendo una cantidad de puntos esparcidos por el escenario y es perseguida por fantasmas que quieren comérsela. Este escenario es una plataforma bidimensional vista desde arriba; a nivel lógico está dividido en celdas de 8x8 pixeles, 28 celdas de ancho y 36 de alto; con bordes predefinidos que limitan los movimientos de los personajes; además existen elementos “power up” que permiten invertir el acoso y por un instante es Pacman quién puede comerse a los fantasmas.

Características: comunes en los fantasmas de Pacman Gameinternal.com nos explica el comportamiento de los fantasmas de Pacman: Estando en una celda predefinida al iniciar el juego, los fantasmas deciden cuál será la siguiente celda a ocupar, calculando para ello una celda objetivo, y la ruta más corta para llegar a ésta, se trasladan a la próxima casilla dentro de la ruta trazada y luego vuelven a hacer los cálculos; es al cambio de celda consecutivo lo que un jugador interpreta como avance.

Métodos de movimiento:

Modo Chase (persecución): Normal, toma la posición de Pacman como parámetro para su persecución.

Modo Frightened (asustado): Se activa cuando Pacman se come un “power up” y consiste en huir de Pacman a una velocidad reducida.

Modo Scatter (dispersión): Se activa para cada fantasma en una esquina distinta del escenario, se abandona el algoritmo normal de movimiento y recorre un camino ya predefinido.

Describiendo el algoritmo de Pathfinding A\*:

“Análisis de Algoritmos Pathfinding”. Este algoritmo es una mejora del algoritmo de Dijkstra y trabaja considerando el escenario como una rejilla con muchas celdas: tenemos el punto de inicio dentro de una y nuestro punto objetivo en otra. El algoritmo consiste en calcular un “costo general” para cada celda que rodee la casilla que se esté analizando, se elige la celda de menor “costo general” y luego esta misma se convertirá en la celda de análisis, teniendo como casilla padre aquélla que acabamos de abandonar. Danilo nos aclara que en este algoritmo los valores de los “costos de movimiento” pueden variar dependiendo de la situación que se esté simulando; pero es evidente que un videojuego base Pacman no es necesario que se haga esta distinción, ya que es igualmente costoso trasladarse de una celda a otra en todo el escenario. Antes de profundizar en este algoritmo, necesitamos entender unos conceptos:

G(N): Es el costo de movimiento al trasladarse de una celda a otra. Este se asigna cuando se está implementando el algoritmo.

H(N): Es el costo de desplazamiento desde la celda analizada hasta la celda objetivo. Este valor se calcula utilizando una heurística especificada al implementar el algoritmo A\*.

F(N): Aquí lo llamamos “costo general” y es la suma H(N) y G(N).

Nodo: instancia de una “clase” que expresa características de una celda: identificador, nodo predecesor, fue analizada o no, G(N), H(N), F(N). En general para este algoritmo hablaremos de nodos en lugar de celdas.

Lista abierta: una lista enlazada donde vamos agregando los nodos que debemos de analizar.

Lista cerrada: una lista enlazada donde están agregados y se seguirán agregando nodos que ya no debemos analizar. En este sentido el mejor camino se busca de esta manera:

1-Añade nodo inicial a lista abierta

2-Añade todos los nodos no transitables a la lista cerrada

3-Ejecutar lo siguiente

4- Busca el nodo con menor F(N) en lista abierta

5- Pásalo a lista cerrada

6- Conviértelo en nodo de análisis

7- Para cada nodo circundante hacer lo siguiente.

8-si no está en lista cerrada, ir a 9 sino 10

9- si está en lista abierta, ir a 13

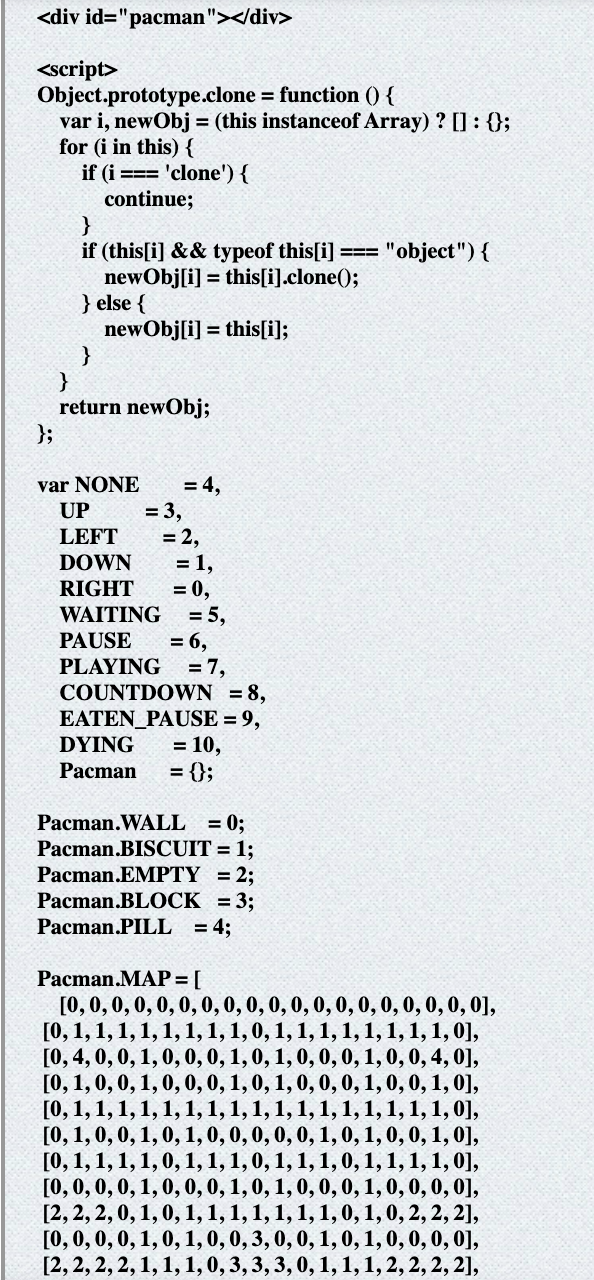
10-agregar nodo a lista abierta

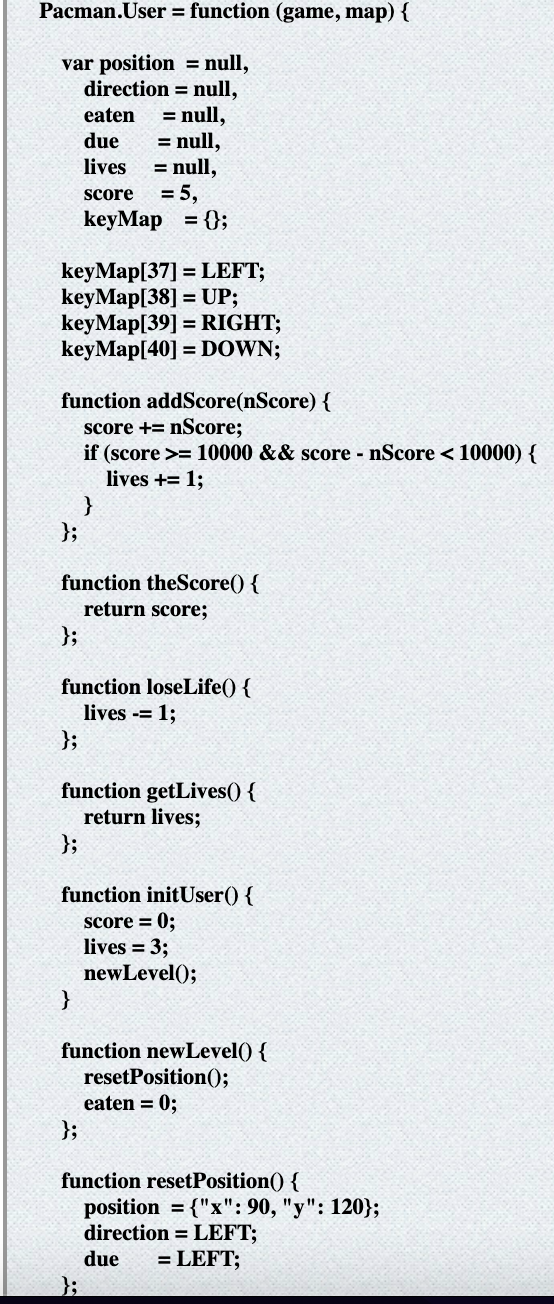
11-Asignarle nodo actual como padre

12-Asignar F(N),G(N),H(N)

13-si está en lista abierta, ir a 14

14-si su G(N) no es el menor, ir a 8





Breve explicación del código:

El algoritmo para los cuatro fantasmas, los movimientos son complejos, ahí está el corazón del juego. Cada fantasma tuviera su propia identidad y sus propios movimientos, para que los cuatro no estuvieran persiguiendo a Pac-man en una sola línea todo el tiempo, que sería algo realmente exhaustivo y plano en un vídeo juego. Por ejemplo el fantasma rojo, Blinky, el si todo el tiempo ataca directamente a Pac-man, lo persigue por todo el laberinto. El segundo fantasma tiene como objetivo posicionarse siempre 3 puntos delante de la boca de Pac-man. Esa es su posición final, entonces, si Pac-man quedaba entre el medio del fantasma A y el fantasma B lo más probable es que Pac-man terminará muriendo, pero como cada uno tiene movimientos independientes con respecto del otro, no pasa siempre. Los otros 2 fantasmas tienen movimientos más aleatorios, cuyo propósito sería explorar el laberinto por todas partes.