Análisis de algoritmos en videojuegos basados en Pacman: "Lo que quiere mi país"

Eduardo José Ávila Portillo*

Resumen:
En el presente artículo se describen, analizan y comparan los algoritmos de movimiento utilizados por los antagonistas de dos juegos muy similares en cuanto a funcionalidad, pero muy diferentes en cuanto a conceptos computacionales aplicados. Pacman aplicando estructuras dinámicas de datos con algoritmos de "pathfinding" y "Lo que quiere mi país" utilizando conceptos más básicos de desarrollo con una dosis de análisis matemático.

Abstract:
In this article are described, analyzed and compared the algorithms of two videogames, same when you play it but so different on computational concepts. Pacman using a pathfinding algorithm and LQQMP using concepts basic of programming and analysis mathematical

1. Introducción

Según Víctor Chambi, autor del artículo "Inteligencia Artificial Juegos": Los fantasmas del juego "Pacman" creado por Toru Iwatani[1] 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[2].

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, ¿será que las búsquedas heurísticas con algoritmos "pathfinding" son el único camino eficaz para desarrollarlo? Mi hipótesis sin comprobación fue un rotundo "¡NO!" y el experimento de comprobación fue el diseñar, desarrollar e implementar un videojuego basado en Pacman, creado desde cero sin utilizar frameworks o alguna API para videojuegos. Esto exige el diseño de los algoritmos de movimiento tanto del protagonista como de sus acosadores. Dicho experimento concluyó con la creación del videojuego "Lo que quiere mi país" ganador del premio al mejor proyecto presentado en el CONINFO 2015, evento organizado por la Asociación Salvadoreña de Profesionales en Computación. En estas páginas acompáñenme a analizar los algoritmos de movilidad tanto en los antagonistas como en el protagonista del videojuego y sacar conclusiones sobre la efectividad de los algoritmos utilizados en relación a las heurísticas ocupadas por el Pacman tradicional.

2. ¿A qué llamamos juegos basados en Pacman?

Pacman es un videojuego de una bola amarilla que avanza comiendo una cantidad de puntos esparcidados por el escenario y es perseguida por fantasmas que quieren comérsela. Este escenario

* *Egresado de Ingeniería en Ciencias de la Computación de la Universidad Don Bosco. Desarrollador web PHP en la Universidad Don Bosco, eduardo.avila@udb.edu.sv.*

es una plataforma bidimensional vista desde arriba; a nivel lógico está dividido en celdas de 8x8 pixeles[1], 28 celdas de ancho y 36 de alto; con bordos predefinidos que limitan los movimientos de los personajes; además existen elementos "power up" que permiten invertir el acoso y por un instante es Pacman quien puede comerse a los fantasmas.

A los videojuegos que cumplan todas (o casi todas) estas características llamaremos: ***basados en Pacman***

3. 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.

a. Modos de movimiento

En Pacman existen 3 modos de movimiento, agrega el artículo de dicha página, es decir formas en que cada fantasma ejecuta su algoritmo de avance. Estos modos son comunes para los 4 de la siguiente manera:

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.

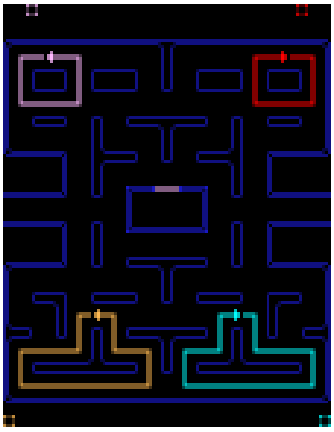

Imagen 1. Rutas predefinidas para los fantasmas al entrar en Modo de dispersión

b. Describiendo el algoritmo de Pathfinding A*

Según nos comenta Danilo Cuevas, en su informe final de proyecto de graduación para optar al título de ingeniero civil en informática: "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 esta en lista abierta, ir a 14
14-si su G(N) no es el menor, ir a 8
15-Asignar nodo actual como padre
16-recalcular F(N),H(N),G(N)
17-si este nodo es la meta, ir a 19
18-este nodo pasa a ser nodo de Análisis, ir a 5
19-guardar el camino
20-recorrer el camino de forma inversa desde el nodo meta hasta el nodo inicial sabiendo cuales son los nodos padres

Resumen de algoritmo A*

Este resumen de algoritmo, es una adaptación de la amplia explicación que Danilo Cuevas Guzmán [4] nos hace.

Aplicado a Pacman, el primer nodo de la ruta es la posición del fantasma, el segundo nodo es la celda a donde moverse.

4. Características individuales en los fantasmas de Pacman

Cada uno posee una "personalidad", la cual (ya hablando de forma técnica) es la celda meta que buscará cada fantasma. Esta característica, junto a los modos de movimiento, logra una aparente persecución colaborativa, aunque en realidad no haya comunicación entre los fantasmas.

a. Fantasma rojo (Blinky)

Utiliza la forma más básica de persecución, el de ruta más corta y su meta objetivo es en sí la posición actual de Pacman, utilizando el algoritmo de pathfinding de manera tradicional.

b. Fantasma rosado (Pinky)

La ruta que calcula está basada en el siguiente paso que dará Pacman, es decir, que para encontrar el mejor camino toma como destino la celda enfrente de la posición de Pacman. Por ello Pinky tiende a desviarse del camino obvio cuando Pacman avanza en la misma línea y en dirección opuesta a él (Ver Imagen 2).

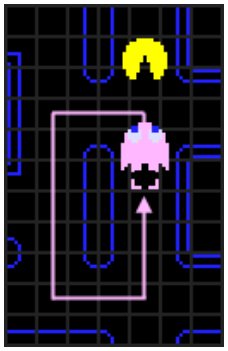

Imagen 2. Pacman está enfrente, la ruta calculada termina atrás de Pinky.

c. Fantasma celeste (Inky)

No comienza a acosar a Pacman mientras éste no haya consumido 30 puntos. Su peculiaridad radica en la casilla que toma como meta. Dicha celda se calcula en tres pasos:

- Identificar la segunda celda en frente de Pacman.
- Calcular la distancia X y la distancia Y desde él hasta dicha celda.
- La celda objetivo está en la posición (2*X, 2*Y) tomando a Inky como (0,0).

Esta nueva celda será la que Inky utilizará como meta al momento de calcular su ruta de movimiento (Ver Imagen 3).

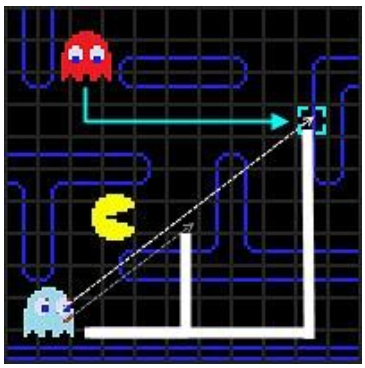

Imagen 3. Inky decidiendo hacia cuál celda debe calcular la mejor ruta.

d. Fantasma anaranjado (Clyde)

Varía su modo de persecución dependiendo de la distancia a la que se encuentre de Pacman; si la distancia es más de ocho celdas, se mantiene en modo Chase, con el algoritmo A* al igual que Blinky. Si la distancia de Pacman es de menos de ocho celdas, Clyde entra en modo Scatter, aunque no esté en su esquina correspondiente. El cálculo de la distancia se realiza cada vez que se desplaza una celda.

5. El videojuego "Lo que quiere mi país"

Es un video juego 2D al estilo Pacman, galardonado como mejor proyecto en el evento CONINFO 2015, organizado por la Asociación Salvadoreña de profesionales en computación, ASPROC. Consiste en un policía que va recogiendo sacos de dinero esparcidos por toda la ciudad, mientras hay 2 delincuentes que lo persiguen a él. El dinero que recolecta pasa a ser parte del presupuesto de seguridad pública y eventualmente dicho presupuesto alcanzará para comprar patrullas policiales que le servirán a nuestro policía para capturar y encarcelar a los delincuentes. Este videojuego ha sido creado para correr en navegadores web, pero sin necesidad de conexión a internet, es decir, con solo tener los script tú ya puedes jugarlo; fue creado en HTML como embalaje, CSS para características visuales como imágenes de fondo y posición de los cuerpos dentro del navegador, además de Javascript como lenguaje para ejecutar procesos y corrida de algoritmos.

Lo que destacaremos en esta oportunidad son los algoritmos de movimiento tanto de Paul (el policía) como de los maleantes que lo persiguen, ya que los movimientos de los últimos mencionados son una automatización del primero.

Imagen 4. Juego "Lo que quiere mi país", ladrones persiguiendo a Paul.

a. Desplazamiento de Paul el policía

Teniendo en cuenta un cronómetro oculto en el juego, Paul es un personaje que cada 0.5 segundos, revisa su bandera de "dirección", cambia de celda y cambia de Sprite, la próxima casilla que ocupara la define dicha bandera, la cual siempre tiene un valor inicial de "Abajo". Este cambio de casilla cada 0.5 segundos es lo que el jugador reconoce como avance o movimiento; la forma en que se controla es bastante simple, cada vez que ocurre un *KeyUp* en una tecla específica, se modifica la bandera "dirección" de esta forma:

KeyUp Tecla A	Hacia la Izquierda
KeyUp Tecla S	Hacia Abajo
KeyUp Tecla D	Hacia la Derecha
KeyUp Tecla W	Hacia Arriba

b. El desplazamiento de los ladrones en "Lo que quiere mi país"

El algoritmo de movimiento es exactamente el mismo que el de Paul, solo se tiene una interrogante: si los antagonistas no pueden ser controlados por el jugador, ¿cómo van a saber hacia dónde moverse? Vimos que los fantasmas de Pacman utilizan un método de "Búsqueda de rutas", pero si esta decisión la estaré tomando cada 0.5 segundos y constantemente mi celda meta estará cambiando de posición, bastaría con concentrarme en cuál es la casilla que me acerca más a mi cambiante celda final, en lugar de calcular toda una ruta; así dejo de lado los algoritmos "pathfinding" y terminé creando un algoritmo que me gusta llamar "paso siguiente", o "next step" para los que gustan el inglés.

Se basa en un análisis numérico muy sencillo: simulamos al villano y a Paul como puntos en el plano cartesiano, con el signo del "Eje Y" invertido y los posicionamos en el cuarto cuadrante. Sus posiciones serán basadas en las que nos describe el top y el left del CSS:

Posición en X = Left
Posición en Y = Top

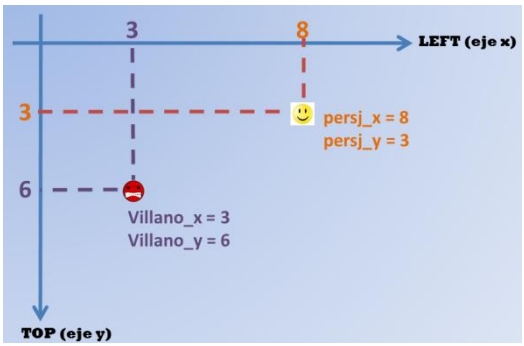

Imagen 5. Simulando al protagonista y al antagonista del juego como puntos en el espacio.

Luego restamos la posición "X" del villano menos la posición "X" del personaje y lo mismo hacemos con las posiciones "Y". Así obtendremos un diferencial de "X" es decir un Dx y un Dy; estos describen la distancia en "X" y en "Y" del ladrón hacia el personaje.

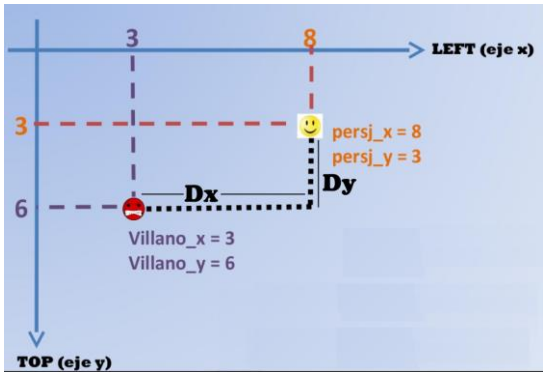

Imagen 6. Grafica de los valores Dx y Dy

Tomando en cuenta que normalmente estos diferenciales no serán

equivalentes, podremos esperar que un diferencial sea mayor que el otro, pudiéndolos así discriminar. Además siendo obvio reconocer que basta reducir una de estas distancias para acercar el villano hacia el personaje, es decir, el siguiente paso que dé el villano reducirá uno de estos diferenciales. Antes de definir un algoritmo debemos responder otra pregunta ¿Cuál de las dos distancias debemos reducir? La regla de "la distancia más corta es la mejor" se ve opacada en esta ocasión ya que si reducimos constantemente el eje con distancia más corta, éste llegará a cero, pero todavía tendremos la distancia en el eje más largo separándonos y el villano nunca tocará a Paul; en cambio al reducir constantemente el diferencial de mayor valor llegará un momento en que ambos diferenciales serán equivalentes y realmente nos habremos acercado a Paul (en esa circunstancia el siguiente paso será elegido aleatoriamente) Teniendo esto en cuenta decidí reducir siempre la distancia más larga.

Ahora, si relacionamos nuestras opciones de movimiento con los ejes coordenados podemos decir que si el mayor diferencial es Dx, nuestras opciones se reducen a izquierda y derecha, en cambio sí es Dy las opciones son arriba y abajo.

Además, tomando en cuenta que en muchas ocasiones los diferenciales nos darán resultados negativos, por lo que una comparación tradicional nos dará un análisis erróneo, si por ejemplo tenemos un Dx=-15 y un Dy=3, gráficamente podremos ver que Dx es el mayor aunque numéricamente no lo sea; por lo tanto, la comparación deberá ser de sus respectivos valores absolutos para saber si trabajaremos con las opciones del eje X o con las de Y.

Elegido el diferencial a trabajar, el signo del mismo indica el valor de la bandera "dirección" de este villano; observando la imagen 6, la coordenada "X" del personaje es mayor que la coordenada "X" el villano y al hacer la resta:

x.villano – x.personaje

Tenemos un Dx negativo, entonces el valor de su bandera será "derecha", como se muestra en la tabla.

Elegido Eje X	Dx > 0	Hacia la Izquierda
	Sino	Hacia la Derecha
Elegido Eje Y	Dy > 0	Hacia Abajo
	Sino	Hacia Arriba

Sintetizamos ideas, en este algoritmo:

```
1-inicio
2-Dx = villano.x – personaje.x
3-Dy = villano.y – personaje.y
4-si abs(Dx) > abs(Dy), ir a 10
5-si Dy > 0, ir a 8
6-direccion = abajo
7-ir a 14
8-direccion = arriba
9-ir a 14
10-si Dx > 0, ir a 13
11-direccion = izquierda
12-ir a 14
13-direccion = derecha
14-Fin
```

Algoritmo "paso siguiente"

Claro, este algoritmo no toma en cuenta los obstáculos, por lo que a veces puede apuntar a una celda inaccesible. Para ello el videojuego usa además un algoritmo corrector que indica el valor de la bandera que no apunte a obstáculos basándose en la dirección previamente elegida:

```
1-inicio
2-si "dirección" es celda accesible, ir a 14
3-si dirección es arriba, ir a  12
4-si dirección es derecha, ir a 10
5-si dirección es abajo, ir a 8
6-direccion=arriba
7-ir a 2
8-direccion = izquierda
```

```
9-ir a 2
10-direccion = abajo
11-ir a 2
12-direccion = derecha
13-ir a 2
14-Fin
```

Algoritmo corrector de dirección

Es con estos dos algoritmos obtenemos que el avance de los ladrones sea totalmente acosador y hasta hostigoso, la cantidad de villanos en el juego es solo dos y ambos utilizan el mismo algoritmo para moverse. La única diferencia es un desfase de 10 ms (milisegundos) que se colocó en uno de ellos, que puede parecer poco, pero después de un tiempo de estar huyendo ese desfase marca diferencia en la posición de los ladrones. Después de jugar varias veces te darás cuenta que los villanos más de una vez lograron acorralarte en un callejón sin salida (como vimos en Imagen 4) por lo que podrías confundirte y creer que utilizando alguna estrategia colaborativa, pero en realidad su persecución es independiente.

c. Modos de desplazamiento de los ladrones

Se reconocen 3 modos de movimiento que surgen tomando en cuenta las posiciones de los villanos, el protagonista y los obstáculos. Con un poco de experiencia, aprendes a controlarlas hasta cierto punto.

Normal:
La separación entre tú y el villano forman un rectángulo y se puede distinguir un lado mayor que el otro. El algoritmo "paso siguiente" no necesita el algoritmo corrector y el villano no deja de perseguirte, es todavía mejor cuando tú y el villano se encuentran en la misma línea, el acoso es imparable.

Vigilancia:
La separación entre tú y el villano forman un cuadrado, básicamente el ladrón queda moviéndose de un lado a otro esperando tu próximo movimiento y elegir cual celda le conviene más.

Huida:
Los algoritmos funcionan al revés, el ladrón huye en lugar de perseguir. Este estado se da cuando Paul obtiene su patrulla policial ("Power up" de Pacman).

6. Conclusión

Se puede crear un Pacman con un algoritmo que no fuera "pathfinding" y que funcione muy bien en los fantasmas. Se puede crear un algoritmo eficaz al utilizar más herramientas matemáticas.

Sin embargo, se resalta que el algoritmo "siguiente paso" deja varios vacíos. Por ejemplo, no considera las celdas inaccesibles y por eso algunas veces necesita un algoritmo corrector que determina la dirección del villano de forma predeterminada, sin importar si te aleja o te acerca a Paul. Deja como posible opción de movimiento la casilla de donde vienes, situación que A* supera haciendo uso de la "Lista cerrada" y estos dos vacíos son los que crean el modo vigilancia, que, en principio no se pensó que estuviera en el juego. No podemos dejar de lado la situación que ocurre cuando Dy=Dx. El algoritmo "siguiente paso" por el momento no logra dar una solución óptima y solo lo resuelve ejecutando las mismas acciones que cuando Dy>Dx. Por lo que podemos decir que aunque los algoritmos de videojuegos sean iguales de eficaces, el A* utilizado en Pacman es aparentemente más eficiente en cuanto a desplazamiento, ya que el único error que genera es el de Pinky cuando huye de Pacman al estar frente a

él; sin embargo al utilizar estructuras como listas y grafos requiere ser implementado en un lenguaje compilado para poder ser eficiente. Salta a la vista que tener que ejecutar el algoritmo A* cada 0.5 segundos en un lenguaje interpretado como Javascript resultaría en un programa lento y bobo, por lo que resulta obligatorio un algoritmo más ágil como el "paso siguiente" para poder crear un pacman en un lenguaje interpretado como en esta ocasión.

Además, este experimento nos ha permitido develar a Pacman como una posible tarea para materias de programación un poco más básicas; teniendo en cuenta siempre el análisis matemático, un alumno de Programación 2 o Programación web ya tiene todas las herramientas necesarias para desarrollar su propio Pacman, pues se demuestra que no es absolutamente necesario aplicar conocimientos de listas enlazadas, grafos, agentes o sistemas expertos para desarrollar juegos de este tipo. Además de ser una prueba contundente de que la matemática, el análisis espacial, gráfico y numérico, son una herramienta fundamental en el desarrollo de software, y será la complejidad del problema que estemos resolviendo quien nos dirá el nivel de teoría matemática que deberemos ocupar, personalmente no me queda más que imaginar la envergadura de los problemas que se pueden resolver si ocupamos los conceptos avanzados de desarrollo de software de la mano con un análisis matemático profundo y muy bien estructurado.

Referencias bibliográficas:

[1] gameinternals.com, (2 Diciembre, 2010) "Understanding Pac-Man Ghost Behavior", citado de internet, del url http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior , el 06 de octubre de 2016.

[2] Chambi Nina, V. A. (2008). Inteligencia Artificial Juegos. Revista de Información, Tecnología y Sociedad, del url: http://www.revistasbolivianas.org.bo/pdf/rits/n1/n1a17.pdf , el 10 de octubre de 2016.

[3] Carlos Soto, M. (2005). Sistema Experto de Diagnóstico médico del síndrome de Guillian Barre. Portal del Sistema de Bibliotecas de la UNMSM, del url: http://sisbib.unmsm.edu.pe/bibvirtualdata/tesis/basic/carlos_sm/cap1.pdf , el 10 de octubre de 2016.

[4] Cuevas Guzmán, D. A. (2013). Análisis de Algoritmos Pathfinding, consultado en línea, del url http://opac.pucv.cl/pucv_txt/txt-5000/UCE5372_01.pdf , el 11 de octubre de 2016.