Instituto Tecnológico de Costa Rica

Lenguajes de Programación – IC4700

Juego de Tanques

Ashley Samantha Acuña Montero – 2021055077.

Warren Ivan Alvarez Huete - 2021072191.

Grupo 50

Oscar Mario Viquez Acuña

II Semestre – 2022

Introducción

Los lenguajes de programación son importantes ya que su función principal es escribir programas que permiten la comunicación usuario-máquina. Unos programas especiales (compiladores o intérpretes) convierten las instrucciones escritas en código fuente, en instrucciones escritas en lenguaje máquina ya sean 1 y 0.

El paradigma lógico, denominado también como programación predicativa, se basa en la lógica matemática. En lugar de una sucesión de instrucciones, un software programado se puede entender como una recopilación de hechos y suposiciones.

La Programación Orientada a Objetos (POO) es un paradigma de programación, esto es, un modelo o un estilo de programación que proporciona unas guías acerca de cómo trabajar con él y que está basado en el concepto de clases y objetos.

En este trabajo se propone crear un algoritmo que pueda buscar el camino más rápido y eficaz entre dos puntos, implementación que se realizará en Prolog, el cual es un lenguaje de programación declarativo, la lógica del programa se expresa en términos de relaciones, representadas como hechos y reglas.

Por otro lado, para la interacción del usuario, se debe de crear una interfaz gráfica para que el usuario pueda jugar los diferentes niveles creados, para lograr está parte se usará Java, el cual es un lenguaje basado en clases y orientado a objetos que está diseñado para tener la menor cantidad posible de dependencias de implementación. Es un lenguaje de programación de propósito general destinado a permitir que los programadores escriban una vez y se ejecuten en cualquier lugar.

Análisis del problema

El problema que se presenta consiste crear un juego de tanques donde por medio de backtracking, busca la ruta más corta entre dos puntos, en este caso, los puntos serían entre el tanque principal y los tanques enemigos, estos tanques enemigos tiene que proteger ciertos objetivos planteados en el lugar de batalla, por otro lado, el tanque principal podrá defenderse por medio de disparos siempre y cuando los tanques enemigos se encuentren cerca de él.

Está búsqueda de soluciones debe de repetirse cada tres movimientos del tanque principal, de manera que los tanques enemigos tengan una nueva ruta de donde se encuentra el tanque principal para eliminarlo.

Los muros en el campo de batalla son importantes ya que ayudan a proteger los objetivos a derribar, además, estos muros no pueden ser derribados por ningún tanque en el juego a realizar.

El tema de vidas, serán nada más tres vidas y lo mismo para los movimientos a realizar el tanque principal, movimientos que solo pueden ser cuatro direcciones básicas, los movimientos de los tanques enemigos tienen que ser definidos y con una velocidad adecuado por el programador

Para la interacción del usuario, se debe de crear una interfaz gráfica donde podrá jugar y realizar diferentes movimientos y disparos con el fin de saber las diferentes maneras de ganar o de perder, cuales son los mejores caminos para poder llegar a los objetivos a derribar.

Para ello se usará un lenguaje que sea orientado a objetos a elección del programador y asegurar que pueda tener una interacción con el lenguaje lógico, que en este caso es Prolog.

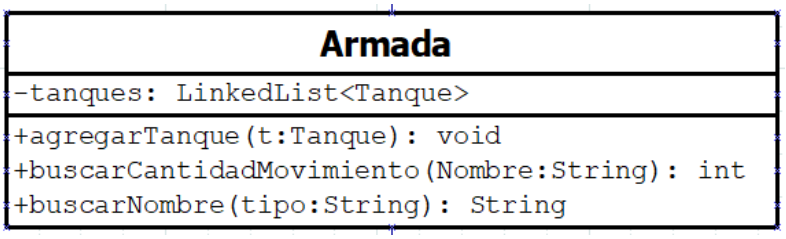
También el usuarios podrá contemplar un botón para iniciar el juego. La finalización del juego a realizar puede estar definida o por el cierre de la pantalla, o por un botón de fin o por las acciones propias del juego.

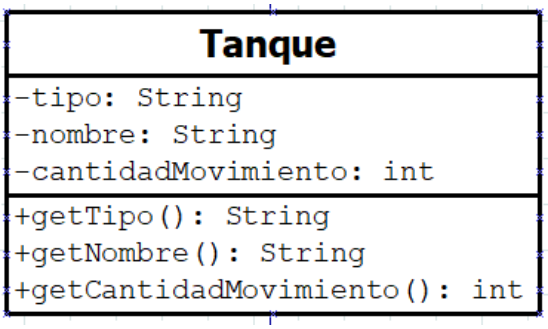
Solución del problema

Para la implementación de este juego de tanques se usó prolog para la búsqueda de soluciones, donde se encarga de buscar una ruta que sea eficaz entre los tanques enemigos y el tanque principal que es controlado por el usuario por medio de la interfaz gráfica.

La búsqueda que realiza prolog, es una búsqueda por anchura, también llamada BreadthFirstSearch en inglés, es un algoritmo usado para recorrer o buscar elementos en una estructura de datos como los grafos. Pertenece al grupo de las búsquedas no informadas, o sea, sin heurísticas. Su procedimiento consiste en ir visitando ciertas posiciones de la matriz con el fin de encontrar la mejor ruta para que los tanques enemigos puedan atacar al tanque principal, que en este caso es el que el usuario controla.

Con Java, se creó una clase que se llama Armada, la cual consiste en crear los tanques y ver cuantos movimientos puede realizar cada tanque, la clase Tanque, tiene como atributos el tipo de tanque, ya sea principal o enemigo, el nombre del tanque y la cantidad de movimientos que puede realizar el tanque dependiendo del tipo.





Para los movimientos de los tanques, existe y una clase llamada Función, está función se encarga poner la matriz principal en ceros, ya que el campo de batalla como tal está hecho por una matriz de 10x5. Ciertas posiciones de la matriz son necesarias para saber donde van los tanques de guerra, los muros, objetivos, y tanque principal.

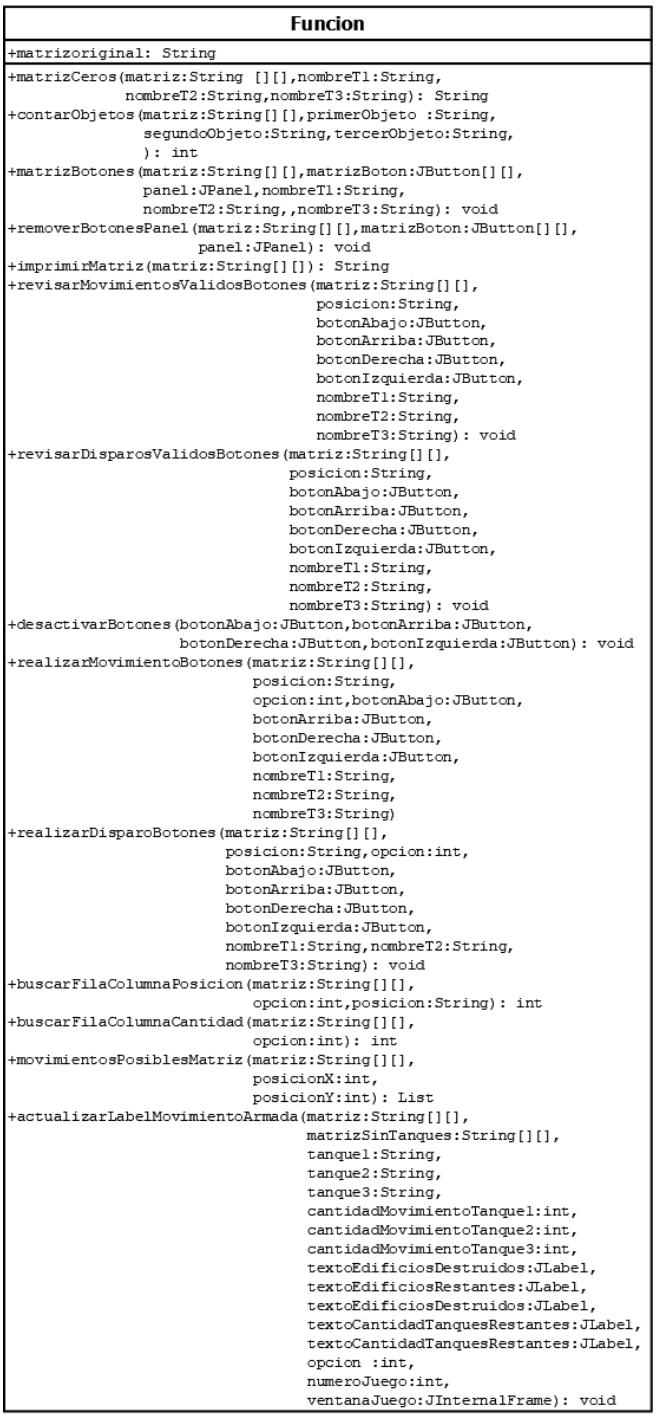
Cada tanque tiene cierta dificultad, en este juego hay tres donde el gris es el A y tiene una dificultad de fácil, el naranja es B y tiene una dificultad de medio, el tanque de color rosa es el C y es de dificultad dificil, por ultimo, el tanque verde es controlado por el usuario, por lo que es el tanque principal a derribar.

También existen funciones con el fin de saber cómo puede funcionar los botones, cuando se pueden activar para disparar a los enemigos, revisar si los disparos son válidos, de igual forma con el movimiento del tanque principal, hasta donde se puede mover, si son válidos los movimientos que realiza y cuáles son los posibles movimientos a realizar.

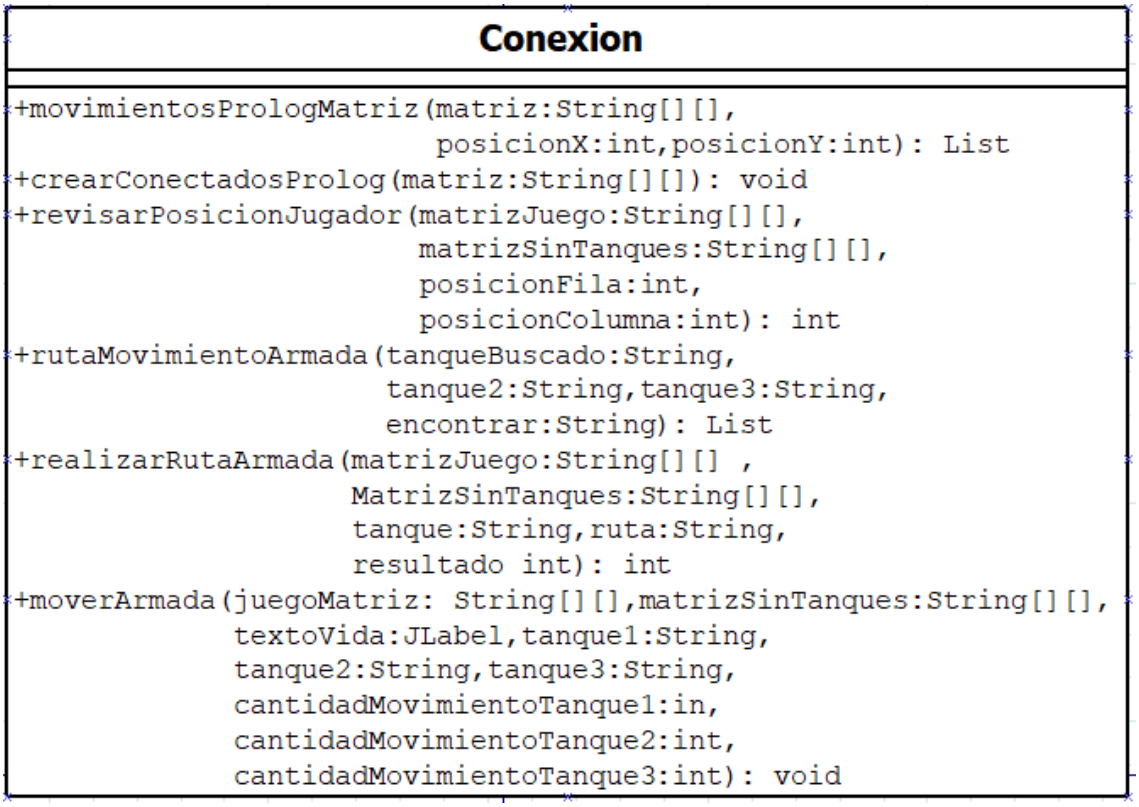
Los movimientos a realizar son siempre tres, ya alcanzando este número, cambia la posición del tanque y las posiciones de los tanque enemigos siguen en lo mismo pero con rutas nuevas de búsqueda para ir en contra del tanque controlado por el usuario.

Con el tema de vidas, el jugador al iniciar solo tiene tres vidas, las cuales se van gastando dependiendo si los movimientos se gastan o si los tanques enemigos lo derriban, en caso de no gastar todas las vidas en el nivel que se encuentra en el juego, son acumuladas para el siguiente nivel, sumándose con las tres vidas que se dan al inicio de cada nivel.

El usuario puede ver los movimientos que tiene, los cuales como anteriormente se van disminuyendo depende de cómo el usuario mueva el tanque, además se busca nuevos posibles movimientos con respecto a los nuevas posiciones de los tanques enemigos ya que ellos sigue permanentes en sus posiciones si fuera el caso de que derriban el tanque principal.



Por último la clase Conexion es la que se encarga de crear la conexión con Prolog y Java, por medio de asserts, Prolog puede saber cual es la mejor ruta entre dos puntos, en este caso, entre el tanque enemigo y el tanque principal, recibe nuevas posiciones cada tres movimientos, verificando por medio de una función, la nueva posición del tanque principal.



Análisis de resultados

A continuación resultados finales del sistema ha implementar

Requerimiento	Estado	Observación
Conexión Prolog y Java	100 %	//
Búsqueda de rutas	100 %	//
3 niveles de juegos	100 %	//
Tanques enemigos y tanque principal	100 %	//
Objetivos, muros y campo	100 %	//

de batalla		
El jugador tiene hasta tres vidas, número de movimientos, dentro de cada nivel para tratar de destruir los objetivos	100 %	//
Botones de disparar y movimientos.	100 %	//
Cada tanque enemigo cuida objetivos, además, calcular la posible mejor ruta para ir hacia el tanque del usuario	100 %	//
Uso de lenguaje lógico y POO	100 %	//
Interfaz gráfica	100 %	//
Se repite la búsqueda por cada 3 movimientos del tanque manejado por el usuario.	100 %	//
Ventana de instrucciones, tres de juego y ventana de finalización	100 %	//
Botones de inicio	100 %	//

Conclusiones

En conclusión, usar prolog para la búsqueda de rutas tiene su dificultad, sin embargo, por su backtracking, se puede decir que la búsqueda en su totalidad es muy eficaz, rápida y segura. Para la conexión entre Java y Prolog, se presentaron ciertos problemas, no obstante fueron solucionados, dando así una calidad de búsqueda de rutas, en este caso búsqueda en anchura.

Además, prolog tiene una forma de procedimiento llamado dinámico, la cual consiste que deben de especificar los predicados se usa alguna vez como argumento a una afirmación o retractación.

Con respecto al uso de un lenguaje orientado a objetos, es más fácil ya que todo se realiza por medio de clases, donde cada clase tiene sus atributos y sus métodos, además, las clases y métodos pueden ser estáticos o abstractos.

Recomendaciones

Investigar bastante sobre Prolog ya que es un lenguaje que no muchos lo conocen, si se sabe que se usa mucho para IA, sin embargo tomar en cuenta que todo se realiza en en hechos, predicados y que para realizar un procedimiento es por medio de la recursividad ya que no existen los loops como tal en este lenguaje.

Además, en java existen las interfaces, las cuales ayudan a que ciertas clases tengan el mismo método que funciona diferente con el fin de que los códigos se vean más agradables y que sean buenas prácticas para un futuro.

Por último, para realizar proyectos así, investigar bien cuáles lenguajes son mejores para poder crear una conexión con Prolog, ya que no es algo muy fácil, sin embargo si se tiene el conocimientos requerido entre prolog y el lenguaje a usar, no seria mucho problema.

Bibliografía

Wikipedia contributors. (2022a, octubre 21). Prolog. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Prolog&oldid=1117409121

Wikipedia contributors. (2022b, octubre 28). Java (programming language). Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Java_(programming_language)&oldid=1118797295

(S/f-a). Makingcode.dev. Recuperado el 4 de noviembre de 2022, de https://www.makingcode.dev/2016/05/metodos-de-busqueda.html

(S/f-b). Bham.ac.uk. Recuperado el 6 de noviembre de 2022, de https://www.cs.bham.ac.uk/research/projects/poplog/doc/prologhelp/dynamic