|  |  |
| --- | --- |
| **Instituto Tecnológico de Costa Rica**  **Ingeniería en Computación**  **Sede Regional San Carlos** | Segunda Tarea Programada  Lenguajes de Programación |
| Prof. Oscar Víquez Acuña. | Tank-Attack |

**Objetivos**

* Aplicar conceptos de Programas de programación Lógico y Orientado a Objetos.
* Generar una herramienta que canalice todos los conceptos estudiados con un cierto nivel de complejidad para generar un reto interesante en el estudiante.

**Descripción del proyecto**

El juego a desarrollar es similar al clásico juego de tanques con algunas variantes en donde existen varios tipos de tanques, el que es manejado por el usuario y otros que cuidan los objetivos que el tanque del usuario debe destruir. Siendo así y con el propósito de explotar al máximo las fortalezas de los paradigmas a aplicar, las siguientes serán las características básicas del juego:

1. El juego tendrá un máximo de **3 niveles** donde la colocación de los objetos del mismo debe realizarse de forma aleatoria en todos los casos. Cada nivel acaba cuando hayan sido destruidos todos los objetivos primarios del mismo
2. Existen **tanques** enemigos que defienden **dos tipos** de objetivos primarios existentes. Estos tanques deben a toda costa defender estos objetivos tratando de destruir al tanque principal manejado por el usuario. Existirán tres tipos de tanques enemigos con capacidades diferentes que serán definidas por el programador. Solo debe existir un tanque que cuide cada objetivo pero dicho tanque puede ser de cualquiera de los tres tipos de tanques. Tanto a cantidad de objetivos, su colocación y el tipo de tanque que cuidará cada objetivo, será definida por el programador, aunque se sugiere tratar de asignar de manera aleatoria para evitar siempre los mismos resultados en los algoritmos de búsqueda.
3. Los tanques que defienden los objetivos solo deben disparar cuando se encuentren “cerca” del tanque del usuario. La definición de “cerca” quedará a criterio del programador
4. Existen tanto a nivel de los bordes como dentro del tablero de juego una serie de **muros** que pueden bloquear el paso de los tanques y de los ataques. Estos muros **no** deben ser destruidos ni por el tanque del usuario ni por los tanques enemigos.
5. El tanque del jugador tiene hasta tres vidas dentro de cada nivel para tratar de destruir los objetivos primarios. Cualquiera de todos los tanques puede moverse solamente en las cuatro direcciones básicas y por ende las balas que dispara, solamente se moverán en las mismas direcciones.
6. Todos los objetos deben mapearse a interactuar con otros objetos cuando sea del caso. No olvidarse que las balas también podrían/deberían ser objetos.
7. Las velocidades de acción de los elementos del juego que no son el tanque manejado por el usuario, deben ser definidas y coordinadas por el programador manteniendo una fluidez que permita que el juego sea llamativo y fácil de utilizar.
8. Debería considerarse la existencia de objetos que controlen el juego y las interacciones entre otros objetos, sobre todo entre los actores “móviles” del juego que probablemente deban programarse utilizando hilos o eventos similares.

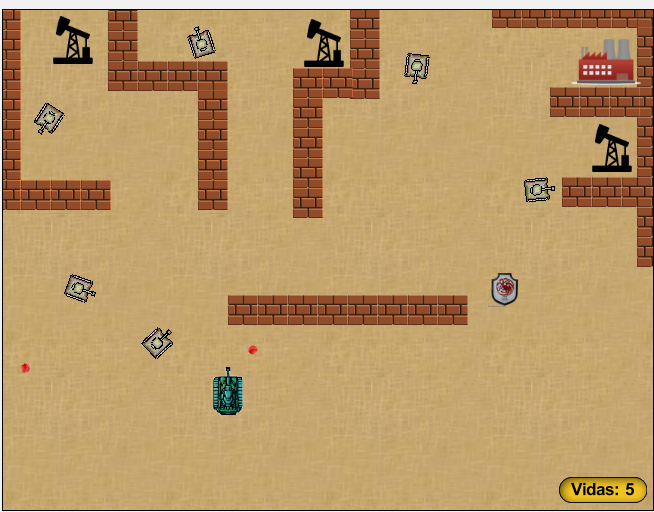


Ilustración 1- Ejemplo de posible implementación gráfica del juego.

Los elementos que se muestra pueden diferir de los indicados en el enunciado

**Búsqueda de soluciones (PRIMER ETAPA A DESARROLLAR EN EL PROYECTO)**

Los tanques que cuidan los objetivos deben moverse hacia donde se encuentra el tanque “agresor”. Esto implica que debe realizarse una búsqueda de un posible camino en la cuadricula de juego que dichos tanques de manera individual deberían tomar. Esta búsqueda de rutas posibles deberá hacerse utilizando el lenguaje de programación Prolog y deberán ser ejecutadas desde el lenguaje de programación Orientado a Objetos que se haya escogido para programar la interfaz y el motor de ejecución del juego de manera gráfica.

La interacción debería ser la siguiente:

1. Desde el lenguaje OO, para cada tanque que cuida objetivos, se solicita calcular la posible mejor ruta para ir hacia el tanque del usuario (el tanque que cuida objetivos debe verse como el inicio de la ruta y el tanque del usuario como el final de la ruta).
2. Se realiza el cálculo para cada solicitud en Prolog, entendiendo que los datos puede o no ser diferentes para cada búsqueda de soluciones y se le “devuelve” dicha ruta calculada al lenguaje OO.
3. El lenguaje OO procede a mover la instancia de tanque para la cual se solicitó la consulta siguiendo dicha ruta
4. Se vuelva a repetir la búsqueda por cada 3 movimientos del tanque manejado por el usuario. La cantidad de movimientos se sugiere, pero al final dependerá del programador con el objetivo de lograr que el juego se desenvuelva de forma fluida.

Para el cálculo de la ruta en Prolog, se solicitará que se implemente una búsqueda en profundidad tal cual ha sido vista en clase, procurando que genere una sola ruta y no muchas para luego obtener la óptima, porque esto será clave para reducir significativamente el desenvolvimiento del juego a nivel gráfico.

Debe tomarse en consideración que al haber varias pantallas en el juego y siendo que la pantalla es la que delimita la cantidad de datos (hechos) que habrá en Prolog, su ingreso a la memoria del intérprete deberá hacerse de manera dinámica (assert).

**Interfaz de usuario**

Se definen los detalles de interfaz de usuario que deben implementarse

1. El juego será desarrollado en cualquier lenguaje de programación OO que el programador desee. Eso sí debe considerarse que debe haber una interacción con Prolog y probablemente esto sea punto de quiebre para la escogencia del lenguaje OO.
2. El juego debe contemplar un botón de inicio del juego. La finalización del mismo puede estar definida o por el cierre de la pantalla, o por un botón de fin o por las acciones propias del juego (que se hayan destruido los objetivos o que el jugador haya perdido todas sus vidas)
3. Debe implementarse un módulo simple que permita editar pantallas. Se sugiere una edición de texto y no necesariamente gráfica, pero al final es su diseño y decisión.

**Detalles adicionales**

Se debe cumplir con las siguientes restricciones:

* Debe diseñarse y documentarse un diagrama de clases para el diseño de los objetos del juego y sus componentes principales (incluyendo métodos principales).
* Si el lenguaje OO tiene las capacidades, deben ocultarse adecuadamente la información de los objetos de manera que se definan claramente qué elementos son privados y cuales otros son públicos.

**Notas Finales para la Tarea:**

* La tarea puede realizarse en grupos de máximo dos personas. Bajo ninguna circunstancia se permitirán ***COPIAS*** de trabajos.
* que ha sido utilizando en otras documentaciones de proyectos del curso.
* Fecha de entrega: lunes 16 de Junio de 2025.
* Hora de entrega: hasta las 10:00 pm a través por el Tec Digital.