

TECNOLÓGICO DE COSTA RICA

Escuela de Computación

***Lenguajes de Programación***

**IV Proyecto Programado**

Andrés García Salas

Raquel Mora Rojas

Oscar Víquez Acuña

Sede San Carlos

12 de noviembre, 2017

# Tabla de Contenidos

[Tabla de Contenidos](#_22stx5jqbs74) 2

[Introducción](#_16p7cd7l60c0) 3

[Análisis del problema](#_fy51rm38p4jt) 4

[Solución del problema](#_6fgcsen7ecvn) 5

[Diagrama de clases](#_fmogkuesqw35) 6

[Análisis de resultados](#_oa4u446w2rkm) 7

[Conclusiones](#_881snrse899i) 8

[Recomendaciones](#_cw1t247xu44t) 8

[Referencias](#_z9a28qst4o38) 9

# 

# 

# Introducción

La abstracción de la realidad es una de las principales ventajas de la Orientación a Objetos, es por ello que este paradigma es uno de los más populares en el mundo de la programación. Por otro lado, la creación de programas sustenta muchas necesidades o facilita instrumentos de entretenimiento/trabajo para los individuos; en esta ocasión se presenta la oportunidad de reconstruir un juego que ha sido olvidado por las nuevas generaciones y que se puede reconstruir por medio del uso de tecnologías como aplicaciones web.

El uso del lenguaje de programación JavaScript combina a la perfección estas dos características, siendo un lenguaje de uso común en la creación de aplicaciones web y con la potencia que permite la orientación a objetos, de forma que el desarrollo del juego del Tanque pueda realizarse de la manera más óptima y que además tenga una apariencia refrescante a la vista del usuario, dejando un portillo para visualizar la creatividad en cuanto a interfaz de refiere de los desarrolladores de la plataforma de juego.

Para la realización de este proyecto es completamente necesario tener conocimientos de nivel intermedio en tecnologías web, orientación a objetos, diseño de interfaz interactiva con canvas y apetito de aprendizaje para determinar la forma de darle solución a este problema en la cual se satisfagan todos los criterios emitidos por el usuario.

Dentro de los apartados de este documento se podrá encontrar un detallado análisis del problema, la forma específica en la que se le encontró solución, un diagrama de clases que detalla la construcción misma del proyecto, un análisis de los resultados obtenidos y por último las conclusiones y recomendaciones respectivas. Para comprender mejor lo que realiza el programa se recomienda leer cuidadosamente el contenido de este documento y a su vez consultar la documentación interna del mismo.

# 

# Análisis del problema

El juego del tanque es un videojuego con muchos años de antigüedad que consta de un tablero rodeado de “paredes” que cumplen la función de borde, que impide que el tanque que controla el usuario, los tanques enemigos y los objetivos no se salgan del rango del tablero de juego; estas paredes son indestructibles, por lo que no importa cuántas balas se le dispare no perecerán. El objetivo primordial del juego es lograr la destrucción de los obstáculos (ya sean paredes o tanques enemigos) que impiden la derrota de los objetivos finales; esto se logra por medio de las balas que el usuario debe disparar desde el tanque que controla para la eliminación de aquello que le impida lograr su cometido.

El usuario debe tener la facilidad de interactuar con el tanque y las balas por medio del teclado con las teclas de arriba, abajo, izquierda, derecha y para disparar se debe contar con la opción de disparar desde la barra espaciadora.

Algunas otras restricciones impuestas para la elaboración del juego son las siguientes:

* 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
* 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.
* Todos los objetos deben tener métodos relacionados con la interacción con otros objetos. No olvidarse que las balas también deben ser objetos.

# 

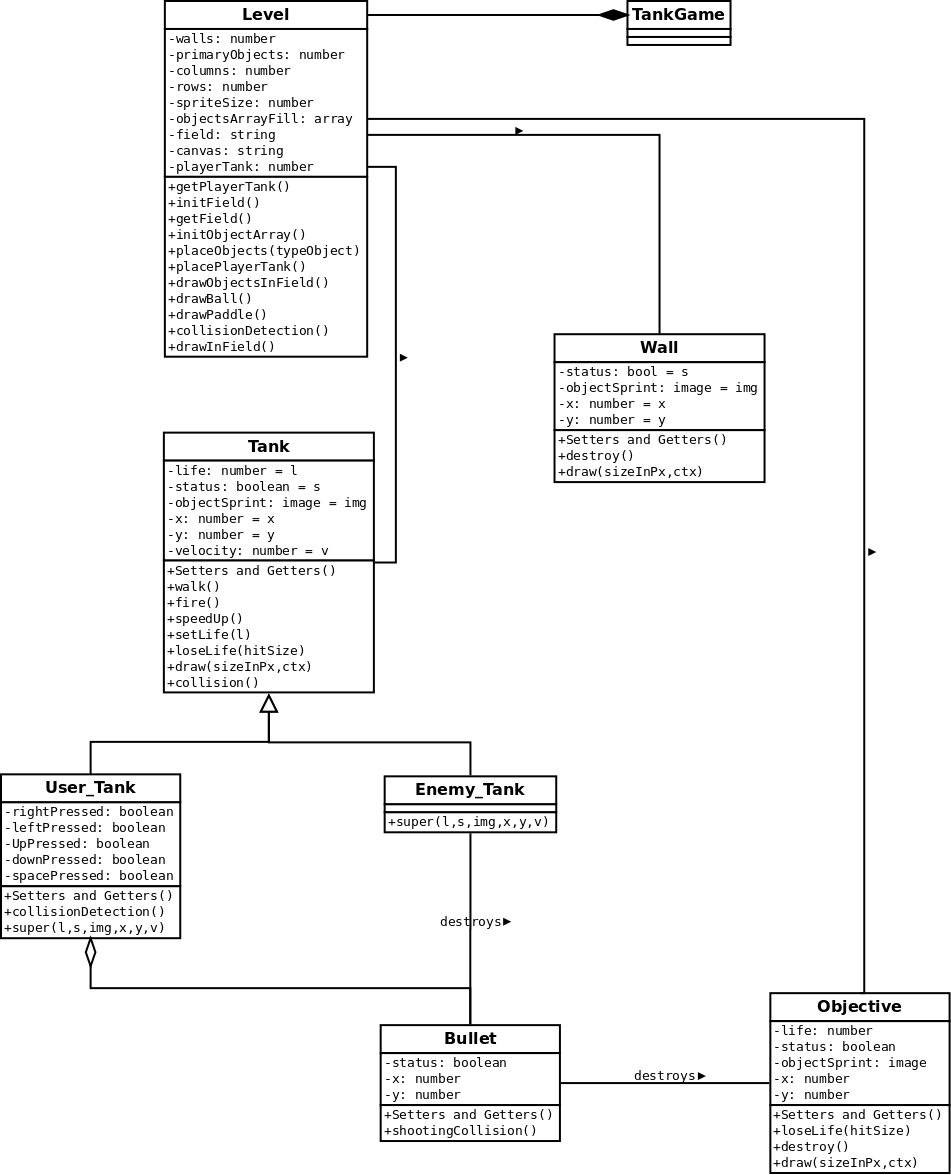
# Solución del problema

Para iniciar con la solución del problema primeramente se debió realizar una pequeña investigación acerca del uso de canvas, de la propiedad setInterval y de cómo manejar POO en javascript (Herencia, Abstracción, etc). Todo esto con el fin de poder establecer las clases y estructuras que se tenían planteadas en el diagrama de clases. Existían otras formas además de canvas para el manejo de la parte gráfica como SVG, sin embargo la primera nos pareció la más adecuada y fácil de entender.

En lo que respecta a las etapas de la creación del juego, la implementación de herencia fue bastante ventajosa, ya que permite ahorrar la duplicación de código sin ninguna necesidad. Con el manejo de objetos cada una de las funciones utilizadas en el juego se pueden entender muy fácilmente. Ya que por su estructura hasta alguien que no programa podría entender un código bien hecho en objetos. Pero bueno volviendo a la solución se empieza inicializando el nivel y todos los objetos que irán en el, primero se colocan los muros y los objetivos primarios ya que el fondo viene colocado con un css, luego de colocar estos objetos se procedió a colocar el tanque del jugador y los tanques enemigos, todos con una posición aleatoria distinta en el nivel.

Una vez realizado todo este primer proceso, es necesario entrar a la segunda etapa donde el juego empieza a dibujarse en el canvas cada 100 milisegundos con el setInterval, acá el proceso se compone de los siguientes pasos en sus respectivos órdenes: limpiar la canvas, actualizar cantidad de primeros Objetivos en el nivel, volver a dibujar los objetos en el canvas (con sus nuevas posiciones, etc..), se vuelven a dibujar los tanques tanto enemigos como el de jugador y además luego de cada uno de ellos, se dibujan cada una de la lista de balas disparadas por cada uno de los tanques existentes en el nivel, lo siguiente es que comprueban las posiciones de todos los objetos en el nivel tanto tanques, balas, objetivos. Para ver si se encuentran en colisión, si es así, se destruyen a menos de que alguno de los objetos encontrados tenga más de 1 punto de vida, de esta manera solo se le baja la vida respecto al impacto y sigue en vida para el siguiente loop, por último se eliminan de las pilas los objetos que tienen un status = false, ya que esto indica que se encuentran muertos o destruidos según sea el objeto, y además se comprueban si ya no existen objetos primarios o si el tanque del jugador no tiene vidas, para de esta manera poder decidir si se ha ganado o perdido el nivel.

Si se pierde o se gana se vuelve a reiniciar el nivel pero con posiciones diferentes, ya que no encontramos un manejo adecuado para poder colocar los demás niveles con el setInterval, sin embargo el aumento de dificultad para cada nivel si es posible según nuestra solución.Diagrama de clases

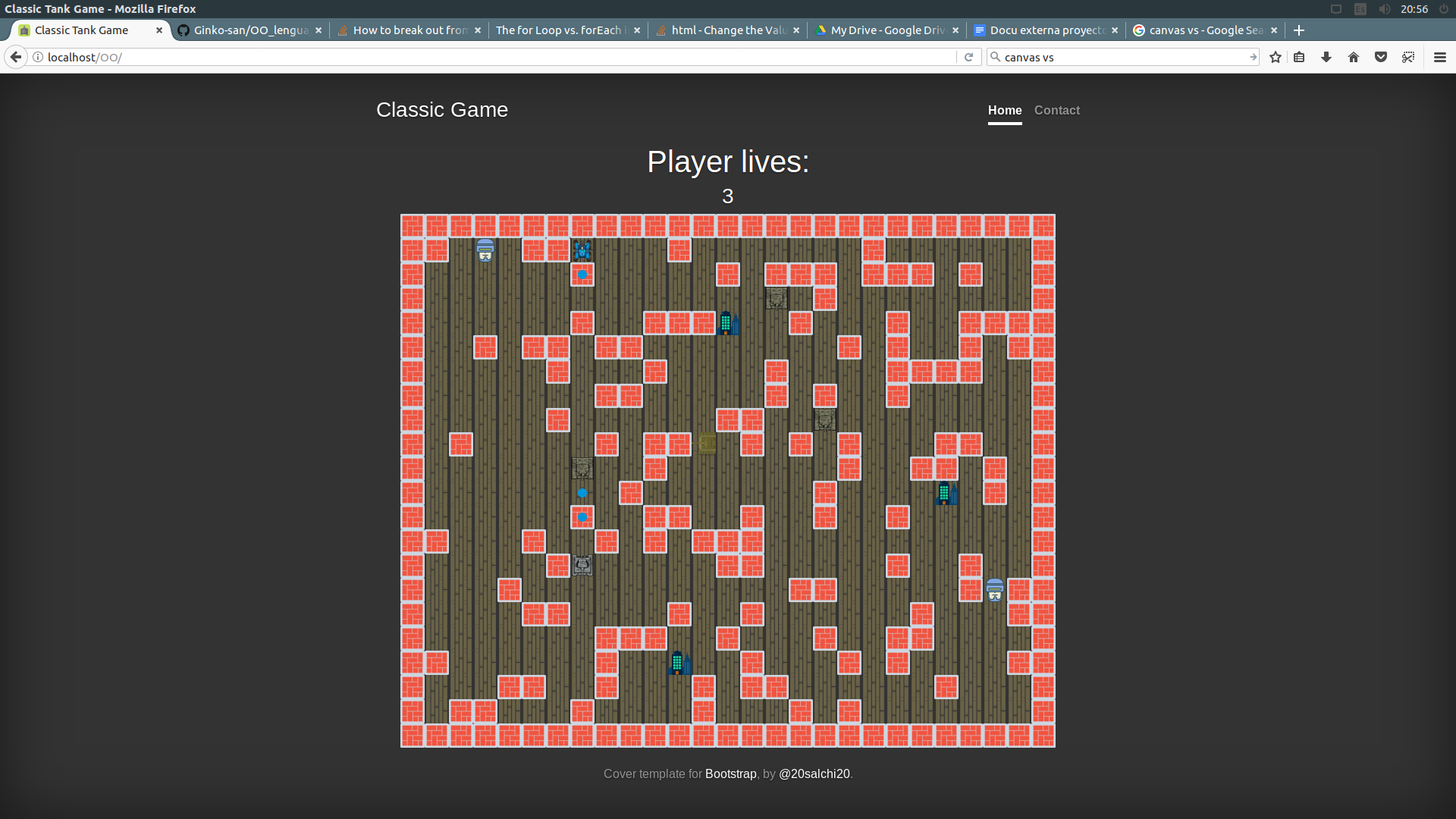


# 

# Análisis de resultados

|  |  |  |
| --- | --- | --- |
| **Requerimiento** | **Estado** | **Observaciones** |
| **Crear tablero inicial** | Completo |  |
| **Posicionar elementos**  **gráficos (tanque usuario, balas,**  **tanques enemigos y objetivos)** | Completo |  |
| **Objetos para cada**  **elemento representado** | Completo |  |
| **2 tipos de objetos primarios** | Completo | Imagenes diferentes,  y uno tiene más vida  que el otro. |
| **Interacción entre objetos**  **por medio de métodos** | Completo |  |
| **Interacción gráfica entre objetos** | Completo |  |
| **Dirección del tanque**  **por medio del teclado** | Completo | Completo las teclas para  manejarlo son las direccionales,  y para disparar  [space + {W,A,S,D}]. |
| **3 Tipos de tanques enemigos**  **con habilidades distintas** | Completo | Uno es resiste mas tiros  que los demás, otro tiene  2 puntos de vida y  pega más fuerte, y el último  es un soldado raso. |
| **Existen muros dentro del nivel** | Completo |  |
| **Los muros pueden ser**  **destruidos por el usuario**  **y no por tanques enemigos** | Completo | Cada muro solo tiene  1 punto de vida. |
| **Tres niveles de juego** | InCompleto | Se pueden crear más niveles  y con distintas dificultades,  solo que no supimos cómo  manejar mejor el setInterval  después de que se ganaba  un juego. |

Previo:



# 

# Conclusiones

Según los resultados obtenidos y reflejados en la anterior tabla es posible entonces concluir que la programación orientada a objetos es perfectamente capaz de darle una solución efectiva a la implementación de un juego de tanques en dos dimensiones. El manejo de clases (objetos), la implementación herencia y el almacenamiento correcto de los datos de manera privada en cada uno de los objetos requeridos por el juego fueron completamente exitosas.

# Recomendaciones

* Leer sobre creación de interfaz gráfica por medio de la herramienta canvas.
* Tener conocimientos intermedios del lenguaje JavaScript.
* Tener mínimo conocimientos intermedios sobre programación.
* Investigar sobre el antiguo juego del tanque.
* Investigar sobre métodos de interacción entre JavaScript y el teclado físico de la máquina.

# Pasos para ejecución

Si usa Linux es necesario que tenga todo el entorno LAMP montado, si es usuario de Windows solo necesita tener XAMP o alguno de esos instalado, el siguiente paso es copiar el proyecto en la “www/html” o “htdocs” según sea. Por último solo accede a la ruta “localhost/nombreDelProyecto” y listo a jugar.

En algunos casos no es necesario tener instalado todo esto, ya que como es js no necesitamos ningún entorno de estos montado, ejecutando el “index.html” en el navegador debería bastar para jugar.

PSDT: Fue testeado siempre en firefox, el diseño es responsive ya que se utilizó bootstrap en la base, sin embargo no damos fe de que funcion en IE -> EDGE.

## 

## 

# Referencias

2D breakout game using pure JavaScript. (n.d.). Retrieved from https://developer.mozilla.org/en-US/docs/Games/Tutorials/2D\_Breakout\_game\_pure\_JavaScript

Array. (n.d.). Retrieved from https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos\_globales/Array

Introducción a JavaScript orientado a objetos. (n.d.). Retrieved from https://developer.mozilla.org/es/docs/Web/JavaScript/Introducción\_a\_JavaScript\_orientado\_a\_objetos

Martini, E. (2009, March 11). POO – Clases Abstractas, Interfaces y Herencia Múltiple. Retrieved from https://emartini.wordpress.com/2008/09/17/poo-clases-abstractas-interfaces-y-herencia-multiple/

Math.round(). (n.d.). Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Math/round

Private Properties. (n.d.). Retrieved from https://developer.mozilla.org/en-US/Add-ons/SDK/Guides/Contributor\_s\_Guide/Private\_Properties

Sa, J. (n.d.). Object-Orientated Design and Programming Unit 9: Abstract Classes and Interfaces. Retrieved from http://www.cems.uwe.ac.uk/~jsa/UMLJavaShortCourse09/CGOutput/Unit9/unit9(0809)/page\_03.htm