VIDEOJUEGO DESARROLLADO

EN RACKET:

TANK SURVIVOR

BECERRA RAMÍREZ CÉSAR ALBERTO - 1744338

ORTIZ CORREA JOSE DAVID - 1740634

VELASQUEZ ACEVEDO JUAN SEBASTIAN - 1744936

PROYECTO

TRABAJO FINAL DEL CURSO

FUNDAMENTOS DE PROGRAMACIÓN

ENTREGADO A:

CASTILLO ANDRES MAURICIO

SANTIAGO DE CALI

UNIVERSIDAD DEL VALLE

PRIMER SEMESTRE

INGENIERÍA DE SISTEMAS (3743)

ÍNDICE **PÁG**

Gráficos

ÍNDICE **PÁG**

1. Introducción

Una de las funcionalidades que cumplen los lenguajes de programación es la de realizar programas que modifican y controlan el comportamiento físico y lógico de las máquinas.
Racket es un lenguaje derivado de Scheme, perteneciente a la familia de lenguajes de programación Lisp, en él se pueden crear e implementar distintos algoritmos funcionales, estos con la ayuda de algunos tipos de datos (como las listas, las estructuras, variables, constantes, entre otros.) se pueden realizar programas que cumplen objetivos predeterminados; como lo puede ser la solución de un problema planteado o la realización de gráficos y plataformas interactivas.

Teniendo como referencia lo anterior en este proyecto se realizó un videojuego utilizando todas las características proporcionadas por Racket, principalmente el paquete de enseñanza Universe, con el cual se pueden crear interfaces interactivas.

2. Justificación

Este proyecto se realizó con el fin de comprobar si un lenguaje de programación funcional, como lo es Racket, y una plataforma como DrRacket son suficientes para la creación de un videojuego, completamente eficaz y funcional, sin la necesidad de utilizar recursos externos. Además, mediante la realización del proyecto se buscó comprobar si los conocimientos obtenidos a lo largo del curso de Fundamentos de Programación fueron suficientes para el desarrollo de un programa óptimo.

3. Objetivos

3.1 Objetivo principal

Aplicar lo aprendido durante el curso de Fundamentos de Programación a través del desarrollo de un videojuego.

3.2 Objetivos específicos

- Realizar el proyecto multimedial utilizando los conocimientos previamente adquiridos del lenguaje de programación funcional Racket.
- Implementar el paquete de enseñanza ‘Universe’, aplicando sus procedimientos estructurales, para ejecutar correctamente el videojuego.
- Perfeccionar y pulir las habilidades adquiridas a lo largo del curso, por medio de la iteración e investigación que se llevará cabo en la realización de este videojuego.
- Desarrollar un pensamiento sistémico que permita entender la relación de todos los componentes funcionales de Racket y la interacción de estos en la creación de una interfaz gráfica.

4. ¿Los objetivos se cumplieron?

4.1 Objetivo principal

El objetivo propuesto se realizó con éxito, en cada uno de los pasos que conlleva la creación del videojuego se utilizaron los conocimientos y conceptos aprendidos a lo largo del curso, tanto en el en aula de clase como en las monitorias suministradas dentro del curso.

4.2 Objetivos específicos

- Este objetivo se cumplió con creces, ya que toda la parte funcional del videojuego se realizó enteramente en la plataforma de DrRacket, utilizando los conocimientos obtenidos en el curso y las monitorias.
- El objetivo se cumplió. Se desarrolló todo el programa teniendo en cuenta los procedimientos estructurales denotados dentro del paquete de enseñanza “Universe”.
- Este objetivo se cumplió a plenitud, para la realización del videojuego se realizó una investigación exhaustiva sobre las diferentes funciones y estructuras que podían ser utilizadas, además de recurrir a las monitorias brindadas por el curso para obtener la accesoria necesaria.
- El objetivo se cumplió. Para la realización y correcta ejecución del programa se necesitó entender a la perfección las diferentes relaciones que existían entre los componentes gráficos del juego y sus componentes funcionales.

5. Marco conceptual

5.1 Lenguaje de programación: Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar, esto con el fin de dar instrucciones a un equipo computacional.

5.2 Cálculo Lambda: Es un formalismo matemático que fue introducido en los 30's por Alonzo Church y Stephen Kleene con el objetivo de dar una teoría general de las funciones, es un sistema formal diseñado para definir funciones, la forma de utilizarlas y la recursión. Consiste en una regla de transformación simple (sustitución de variables) y un esquema para definir funciones.

5.3 Programación Funcional: La programación funcional, o, mejor dicho, los lenguajes de programación funcionales son aquellos lenguajes donde las variables no tienen estado, no hay cambios en estas a lo largo del tiempo y son inmutables. El paradigma de la programación funcional se ha creado explícitamente para permitir un enfoque puramente funcional de la resolución de problemas.

Todo se procesa usando recursividad (un proceso basado en su propia definición) y funciones de alto orden. Esto se debe a los fundamentos matemáticos de la mayoría de los lenguajes funcionales, principalmente con bases en el sistema formal diseñado por Alonzo Church para definir cómputos y estudiar las aplicaciones de las funciones llamado Cálculo Lambda.

El cálculo lambda es universal porque cualquier función computable puede ser expresada y evaluada a través de él. Por lo tanto, es equivalente a las máquinas de Turing. Sin embargo, el cálculo lambda no hace énfasis en el uso de reglas de transformación y no considera las máquinas reales que pueden implementarlo. Se trata de una propuesta más cercana al software que al hardware.

5.4 Paradigmas de programación: Un paradigma de programación indica un método de realizar cómputos y la manera en que se deben estructurar y organizar las tareas que debe llevar a cabo un programa. Es decir, un modelo para resolver problemas computacionales. Los lenguajes de programación, necesariamente, se encuadran en uno o varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar, algo que tiene una relación directa con su sintaxis.

5.5 Racket: Es un lenguaje de programación funcional multiparadigma, creado especialmente con un entorno de desarrollo pedagógico a partir del lenguaje de programación Scheme que a su vez proviene de Lisp. Uno de sus principales objetivos tras su diseño es posibilitar la creación de nuevos lenguajes o dialectos.

5.6 DrRacket: Es un entorno de desarrollo integrado programado en Racket, que facilitará la tarea de programar en Racket. También ofrece raco, una herramienta para la línea de comandos que permitirá instalar paquetes o compilar librerías.

El editor proporciona resaltado de origen para sintaxis y errores de tiempo de ejecución, organización en paréntesis, un depurador y sistema metódico algebraico. Sus características amigables para los estudiantes incluyen soporte para múltiples niveles de lenguaje (Estudiante Principiante, Estudiante Intermedio, etc.). También tiene soporte de biblioteca integrado y sofisticadas herramientas de análisis para programadores avanzados.

Este entorno se caracteriza por contener distintas herramientas que otorgan una gran cantidad de funcionalidades, en este caso se definirán las correspondientes a la creación de interfaces gráficas y manejo de físicas las cuales serán primordiales para el cumplimiento de los objetivos de este proyecto:

5.6.1 Datos: Los datos son representaciones simbólicas de atributos o variables, tanto cuantitativas como cualitativas. También se puede entender un dato como la representación de la información que es manipulada por el programador con el fin de lograr desarrollar la solución al problema que se le plantea.

En Racket los datos pueden ser:

Datos atómicos (datos que no se componen de otros, que no se pueden dividir en sub-datos) como lo son números, booleanos, símbolos o datos compuestos (datos que están compuestos por datos atómicos) como estructuras, listas y cadenas.

5.6.2 Funciones: Las funciones en Racket son puramente matemáticas (dependencia entre dos cantidades de variables), en las que se verifican ciertas propiedades como la transparencia referencial (el significado de una expresión depende únicamente del significado de sus subexpresiones) y, por tanto, la carencia total de efectos colaterales.

Es un conjunto de líneas de código que realizan una tarea específica, y que a partir de una o más variables puede retornar algún tipo de dato. Las funciones tomar parámetros que modifiquen su funcionamiento. Esta puede ser invocada o ejecutada las veces que se necesiten.

5.6.3 Estructuras: Datos complejos que representan objetos compuestos de varios valores simples, determinando sus propiedades las cuales son piezas de información. Una estructura combina un número fijo de valores en una sola pieza de datos.

5.6.4 Listas: La lista es una estructura de datos fundamental de racket, en esta se puede almacenar variedad de elementos, incluyendo otras listas o estructuras, el número de datos que contiene una lista no está definido, por lo tanto, se puede tener infinidad de estos. Si una lista no tiene ningún valor es llamada “empty”.

5.6.5 Mundo: Es un contenedor de que guarda todo lo susceptible a cambios, es decir, los datos de cada cosa que interactúa en el mundo y puede ser afectada, cambiando sus atributos. Un ejemplo de mundo es el videojuego de tanques, el cual almacena la posición de cada bala, así como una puntuación, vidas, etc.

5.6.6 Universo: Es un contenedor de mundos, que gestiona cómo interactúan entre ellos y cómo se deben desarrollar, a partir de funciones que organizan los eventos al dispararse. Este nace a partir del bing-bang.

5.6.7 Funciones que manejan eventos: Al llamar a la sentencia bing-bang, esta recibe como parámetros diferentes tipos de funciones que vienen preestablecidas por el paquete y denotan todos los eventos que se ejecutan durante el juego. Estas reciben como parámetro las funciones que controlan cómo se manifiestan los eventos y cómo interactúan con el mundo.

5.6.8 Polimorfismo: Es el proceso de combinar dos o más funciones similares con el fin de mejorar el rendimiento del programa, depurar errores y realizar más de una tarea con la misma función.

5.6.9 Definiciones Locales: Esta herramienta permite agrupar una secuencia arbitrariamente grande de definiciones que se relacionan entre sí de forma similar a las que se encuentran en la ventana de Definiciones, es de gran ayuda puesto que le da organización al código fuente y en algunos casos, permite optimizar ciertos procedimientos.

5.7 Inteligencia artificial: Es la capacidad de una máquina de percibir su entorno y llevar a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea.

Antes de empezar a codificar y hacer uso de estas herramientas se debe tener en cuenta, el siguiente algoritmo para la solución de un problema. Primero, se debe identificar el mundo del problema, luego se deben determinar sus componentes, después se procede a la creación del mundo y de los eventos que van a cambiar su estado, y por último se define cómo será la interfaz gráfica.

6. Descripción

El juego presentado se basa en que dos tanques, cada uno operado por un jugador diferente, se enfrentan a una horda de tanques enemigos los cuales deben eliminar para obtener puntos. El objetivo final del juego es tratar de alcanzar el puntaje más alto posible. Este último aparte de mostrar el rendimiento de los dos individuos a lo largo de la partida funciona como medidor de la dificultad del juego, ya que al aumentar el puntaje cambian algunas variables del juego como lo son la cantidad de tanques enemigos, su velocidad de disparo y de movimiento.

Los tanques enemigos poseen solo una vida, por lo tanto, al recibir el impacto de un proyectil este desaparecerá y volverá a salir en uno de los puntos de generación de enemigos que se encuentran distribuidos en el mapa, no obstante, se debe tener en cuenta que un disparo por atrás no elimina a los enemigos, esto con el objetivo de que la dificultad del juego sea mayor.

A diferencia de los enemigos cada jugador posee tres vidas, esto quiere decir que puede ser impactado por tres proyectiles antes de desaparecer, en el momento en el que los dos jugadores pierdan todas sus vidas el juego acabará.

La interfaz gráfica diseñada para este videojuego es la siguiente:

Grafico 1, Interfaz

La interfaz puede dividirse en dos secciones, en la zona ubicada a la izquierda se puede observar el mapa de juego, este consta de distintas paredes que imposibilitan el paso en algunas zonas, haciendo que los jugadores tengan que movilizarse en un camino predeterminado, el cual está construido en forma de laberinto. En la segunda sección se puede identificar el logo creado para el juego y el espacio en el que se aloja la información referente a los puntajes y vidas de los jugadores. Los récords más altos de cada partida son almacenados en un archivo a parte para desplegarlos después en la interfaz.

Las características del tanque se pueden dividir en su versión gráfica que es representada con una imagen y otra física que se materializa como un círculo con el fin de facilitar su movimiento y la interacción con otros objetos del juego. El primer tanque es manejado a partir de las teclas A, W, S, D, siendo, izquierda, arriba, abajo y derecha respectivamente y dispara mediante la presión de la tecla SPACE. El segundo tanque, realiza su movimiento a partir de las flechas y dispara con la tecla P.

El funcionamiento de los tanques enemigos se ejecuta mediante un proceso temporal que determina el movimiento de estos, permitiendo así que los tanques se desplacen de manera automática a través del mapa.

Por otra parte, el movimiento de las balas se controla a partir de la dirección en la que apunta el tanque que dispara y el paso del tiempo.

7. Desarrollo

7.1 Desarrollo lógico

Para el desarrollo del juego se utilizaron varias características proporcionadas por DrRacket, implementando herramientas conocidas y trabajadas a través del curso de Fundamentos de Programación. Entre estas herramientas están, recursividad, composiciones de listas y estructuras, que ayudan a la organización de los datos para un uso óptimo en las funciones requeridas; funciones aritméticas predefinidas, que permiten la maleabilidad de constantes y variables. Lo anterior posibilita una relación armónica entre las primeras definiciones del código que combinan dimensiones y diseño del mapa, estados y vidas de los tanques, imágenes (utilizando el paquete de enseñanza 2htdp/image) y velocidades.

También es primordial la utilización de algunos paquetes de enseñanza proporcionados por el editor, entre ellos el principal es 2htdp/universe que implementa conceptos como funciones que manejan eventos, Mundos y Universos en el BigBang.

Una de las funciones principales para manejar eventos del teclado es el ‘on-key’, en la que se controla el movimiento de los usuarios, modificando las direcciones posibles (“up”, “down”, “left”, “right”) de los tanques y la acción de disparar una bala aliada.

A fin de ejecutar una serie de funciones dependientes de un transcurso temporal, se hace uso de la herramienta ‘on-tick’, con la que se evalúan distintos parámetros que retornan un mundo con atributos que se van actualizando a medida que se cumplen ciertas condiciones, por ejemplo, las vidas de los jugadores, el movimiento de los enemigos y el de las balas.

Un aspecto destacable en Tank Survivor es la Inteligencia Artificial de los enemigos, esta se basa en que el enemigo cuando va a cambiar de dirección para ejecutar su movimiento tome el camino despejado en el que no colisione con ningún otro elemento del mundo. Entrando en detalle, cada enemigo evalúa si en su siguiente movimiento impacta con otro elemento, y si este es el caso, entonces tomará otra dirección hasta que encuentre una carretera para su libre movimiento. Dado el caso que tenga bloqueadas sus 4 direcciones posibles, este seguirá buscando una posible hasta encontrarla.

Con el objetivo de almacenar y desplegar los puntajes más altos, se hizo uso del paquete 2htdp/batch-io, que permite leer y escribir ficheros externos a través de simples líneas de código. Tank Survivor se vale de un archivo con extensión .csv, y cada vez que una partida finaliza compara el puntaje obtenido con los almacenados en este, para después sobrescribirlo con los cinco más altos en orden descendente.

Otro paquete de enseñanza utilizado fue 2htdp/gui, que permite la creación de paneles gráficos y elementos para su interacción con el usuario, como lo son: botones, cajas de texto, etc. Utilizando estas herramientas se creó una ventana de bienvenida, una para las instrucciones y otra que despliega las cajas de texto que utilizan los usuarios para asignar el nombre deseado a cada tanque.

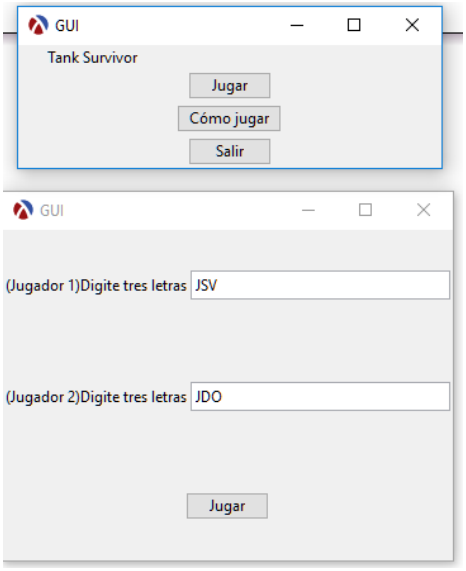

Grafico 2, Ventanas

Por último, cabe destacar que se utiliza el polimorfismo y definiciones locales con el objetivo de optimizar el código en cuestiones de eficiencia y desarrollo.

7.2 Desarrollo Gráfico

Para la realización de la interfaz, el diseño de las paredes, de los tanques y los proyectiles se utilizó una herramienta de edición online llamada Piskel. En ella se pueden crear distintas imágenes con un estilo pixelado que recuerda a los juegos clásicos de las décadas de los ochentas y noventas. Cada objeto del juego fue desarrollado individualmente, y colocado dentro de la interfaz por medio del paquete de enseñanza 2htdp/image.

A continuación, se mostrará un pantallazo del juego en el que se puede evidenciar todas las gráficas que están incluidas en él.

Grafico 3, Screenshot

8. Conclusión

De la realización de este trabajo se pudo concluir que DrRacket es una plataforma que permite la creación de programas interactivos de manera eficiente, aunque para poder realizar dichos programas primero se necesita tener claros algunos conceptos (El polimorfismo, definiciones locales, recursividad, los distintos tipos de datos, composición de listas y estructuras, variables, constantes, implementación de paquetes de enseñanza necesarios para desarrollar interfaces interactivas, entre otros) que permiten su realización y ejecución de manera correcta. Además, se confirmó la importancia que tiene la documentación para lograr un entendimiento adecuado del código realizado.

Por último, se logró comprender el gran trabajo que hay detrás de la realización de un videojuego, desde su aspecto lógico hasta el gráfico.

9. Referencias

- Anonimo. *Beautiful Racket.*. Recuperado de: https://beautifulracket.com/explainer/lists.html
- *Data is the new oil.* Lugar de publicación:Spotless data. Recuperado de: https://spotlessdata.com/blog/data-new-oil
- Urbina. (noviembre 2010). *Lenguajes de Programación.* Lugar de publicación: Blogspot. Recuperado de: http://ceciliaurbina.blogspot.com.co/2010/11/calculo-lambda.html
- Toonders. *Data Is the New Oil of the Digital Economy.* Lugar de publicación: Wired. Recuperado de: https://www.wired.com/insights/2014/07/data-new-oil-digital-economy/
- Anónimo. (noviembre 2017). *CCM*. Recuperado de: http://es.ccm.net/contents/304-lenguajes-de-programacion
- Diaz. (noviembre 2012). *Programando*. Recuperado de: http://www.programando.org/blog/2012/11/que-es-la-programacion-funcional/
- García. (octubre 2016). *Paradigma digital*. Recuperado de: https://www.paradigmadigital.com/dev/la-programacion-funcional-deberias-usarla
- Vaca. (2011). *Paradigmas de progrmación.* Recuperado de: https://www.infor.uva.es/~cvaca/asigs/docpar/intro.pdf
- How to Design programs – Recuperado de: http://www.htdp.org/
- The Realm of Racket – Recuperado de: http://www.it-ebooks.info/