Videojuego para el Aprendizaje de las Estructuras de Datos

Juan Sebastián Cabezas Mateus, Raúl Mauricio Peña Losada, Juan Diego Ramírez Lemos, Santiago Rodríguez Vallejo.

No. de Equipo Trabajo: {*02*}

I. INTRODUCCIÓN

El presente documento define al detalle el diseño de un proyecto en el que se apliquen los conocimientos adquiridos en cuanto a las estructuras de datos. Dicho proyecto consiste en la creación de un videojuego para que personas de todas las edades aprendan de forma interactiva los temas en la materia. Primero se pasa por las motivaciones para desarrollar el proyecto, luego se definen sus funcionalidades y características principales para así mostrar una visualización preliminar. Se explicará cómo es que este videojuego hará uso de las diferentes estructuras de datos y se demostrará cómo es que efectivamente funcionan mediante una serie de comparaciones bajo operaciones de ciertas complejidades. Por último, se explica a profundidad el proceso de desarrollo a nivel técnico y se enlistan las dificultades presentadas, así como las lecciones aprendidas.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

La tecnología computacional se convierte poco a poco en un área del conocimiento clave para la competitividad en el mundo digital. Aprender y aplicar los conceptos que esta área comprende debería ir mucho más allá de solo presentarlo a los estudiantes como herramientas teóricas que probablemente no sabrán en qué situaciones y de qué formas pueden llegar a aplicarlas. El aprendizaje de las estructuras de datos hace parte de esta teoría computacional que puede estructurarse de forma tal, que sea natural adquirir y aplicar el conocimiento por parte de escolares tanto jóvenes como adultos.

Se han planteado diversas maneras de dictar este curso de manera amigable. Se sabe debido a varios estudios que una de las mejores formas para aprender son a través del juego y la didáctica [1]. La creación de un videojuego que utilice conceptos y mecánicas similares a las utilizadas en el curso puede ayudar en el aprendizaje de las estructuras de datos familiarizando a los jugadores con el pensamiento lógico para dichas estructuras.

Como un panorama previo de la efectividad del proyecto, se encuentra que, por ejemplo, en los estudios de Dicheva-Hodge [2] se realizó un minijuego para enseñar las listas de manera didáctica, concluyendo en una encuesta a los estudiantes donde la mayoría dice que el juego les fue de ayuda para su aprendizaje.

Además, Ramle et al [3] presenta la idea de huir de zombies apilando y desapilando rocas, como se haría en una lista de pilas y colas convencional, dando como resultado que, a excepción de un estudiante, todos obtuvieron resultados positivos en una prueba acerca de este par de temas.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

El programa está enfocado a cualquier usuario que desee aprender las bases lógicas de las estructuras de datos de forma sencilla e intuitiva a través de minijuegos.

Todos los usuarios que empleen el programa tienen el mismo rol: Jugador y tienen el mismo acceso a todas las funcionalidades explicadas en la sección V.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

Se encuentra que el producto final será simple y altamente intuitivo para el usuario; esto incluye una extensión corta de las funcionalidades a las que se pueden acceder debido a que solo se busca una interacción directa y concisa con los objetos en el juego y esto implica que la mayoría de acciones sean realizadas mediante eventos (o clics) y solo se tendrá la posibilidad de ingresar datos por teclado al registrar información básica al estilo arcade [4]. A su vez, mediante código se tiene la posibilidad de ingresar datos al programa, pero este es un acceso restringido al público y solo se recurrirá a este medio por razones experimentales.

La pantalla principal que se presenta al usuario será al estilo de un menú de juego arcade, esto incluirá las siguientes operaciones:

- Iniciar juego:
  Saldrá de la ventana de menú y mostrará una malla de niveles de los cuales el usuario puede elegir a cuál acceder.

- Mostrar créditos:
  Mostrará una pequeña ventana con la información de los creadores del juego.

- Guardar juego:
  Efectuará acciones internas del programa para guardar localmente datos de estado de los niveles, así como los puntajes e información suministrada.

- Cargar juego:
  Permitirá establecer todos los parámetros necesarios al estado exacto en la última vez que el usuario guardó su partida.

- Salir del juego:
  Mostrará una pequeña ventana de verificación en la que el usuario decide si efectivamente desea cerrar el juego y en caso afirmativo cierra el programa.

Al ejecutar la operación de iniciar juego, el usuario tendrá la opción de elegir uno de varios niveles. El diseño de cada nivel es acorde a que el usuario pueda aprender y aplicar diferentes conceptos de las estructuras de datos mediante escenarios distintivos y altamente gráficos.

Dentro de cada nivel, sin excepción el usuario cuenta con la posibilidad de ejecutar las siguientes operaciones:

- Pausar nivel/Mostrar menú:
  La operación se ejecuta mediante un pequeño botón ubicado en una esquina de la ventana. Al dar clic, abrirá una ventana que permita al usuario ver información básica del nivel, ir al menú principal, ir al menú de niveles, visualizar su puntaje actual, reiniciar o reanudar su nivel.

- Reiniciar nivel:
  La operación se ejecuta mediante un pequeño botón ubicado en una esquina de la ventana. Al dar clic, todos los elementos en el nivel vuelven a su estado inicial.

- Mostrar información:
  La operación se ejecuta mediante un pequeño botón ubicado en una esquina de la ventana. Al dar clic, abrirá una ventana dentro del juego en la que muestre información importante sobre el nivel actual.

- Confirmar entrada:
  La operación se ejecuta mediante un botón de acción con un nombre distintivo dependiendo del nivel. Al dar clic, el programa hará una comprobación de que el usuario haya cumplido con su objetivo, en caso afirmativo lanzará un mensaje de felicitación, mostrará su puntaje y le permitirá acceder al siguiente nivel. En caso negativo lanzará un mensaje de fallo que le permita reiniciar el nivel.

- Cambiar de nivel:
  La operación puede ejecutarse tras abrir la ventana de menú de pausa de nivel y se mostrará como un botón. Tras dar clic volverá a la malla de niveles de todo el juego. También, esta operación se ejecuta si el usuario pasa un nivel y desea cambiar al que se encuentra inmediatamente después.

- Registrar datos personales:
  La operación se puede realizar tras pasar con éxito un nivel. En dos barras de entrada podrá ingresar su nombre de usuario y edad.

- Visualizar su puntaje y un historial de marcas:
  La operación de visualizar el puntaje registrado por el usuario se realiza implícitamente tras abrir el menú de nivel o tras pasar un nivel. En cualquiera de estas dos situaciones, el usuario tendrá además la posibilidad de visualizar un historial de marcas o récords registrados en el programa, para esto solo debe hacer clic a un botón específico.

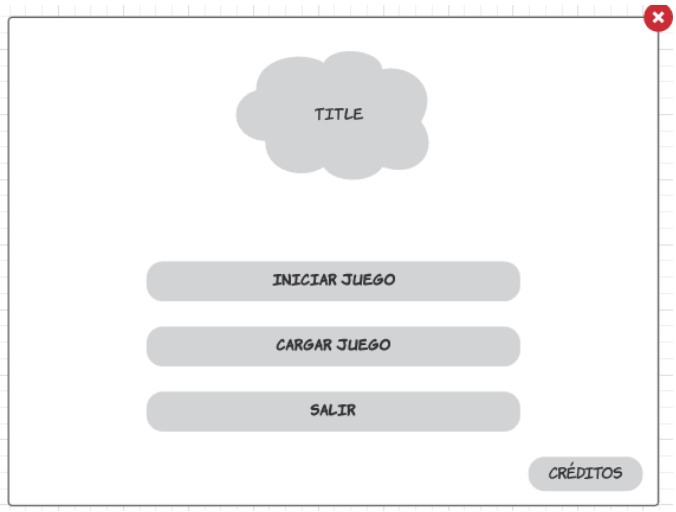
Aparte, dependiendo del nivel, tendrá la opción de mover objetos y oprimir botones especializados. También contará con la posibilidad de regresar acciones.

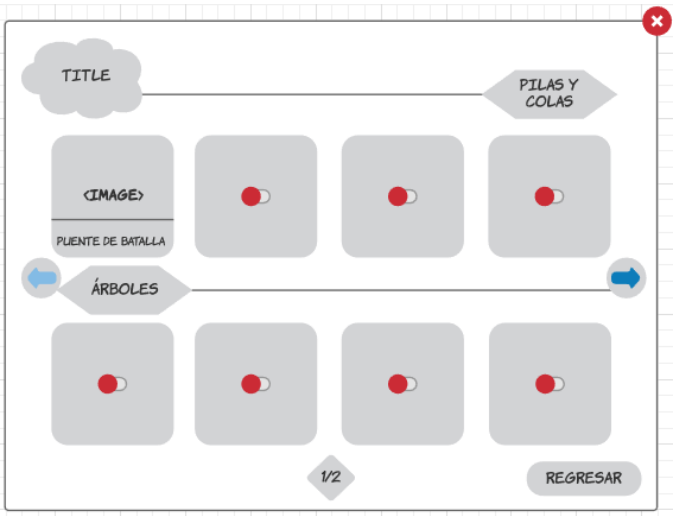
Todo el programa interno se basará en crear y manejar la interacción adecuada entre distintos objetos, así como gestionarlos visualmente mediante el renderizado, ocultamiento y borrado pertinente. La retroalimentación de algunas operaciones se mostrará mediante ventanas pop-up y algunas animaciones básicas. Todas las posibles acciones del jugador están realmente controladas mediante dichas ventanas, limitaciones de uso o restricciones de acceso temporales.

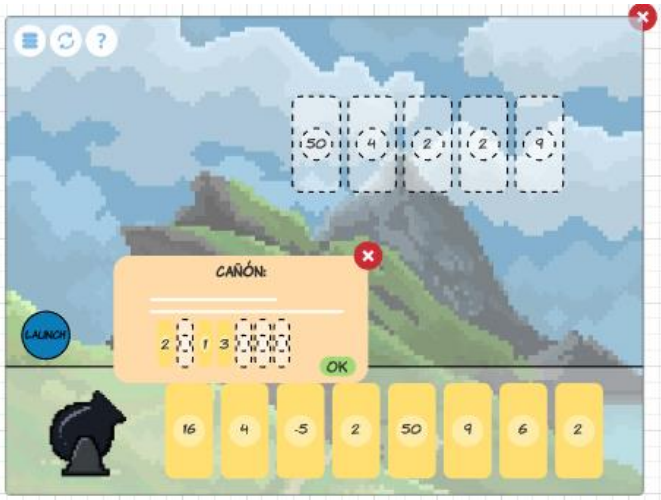
V.DESCRIPCIÓN DE LA INTERFAZ DE USUARIO PRELIMINAR

La previsualización de la interfaz estilo Mockup se desarrolló con la ayuda del aplicativo Lucid Chart. [5]

A continuación, se muestran algunas de las vistas que incluye:







VI. Entornos de desarrollo y de operación

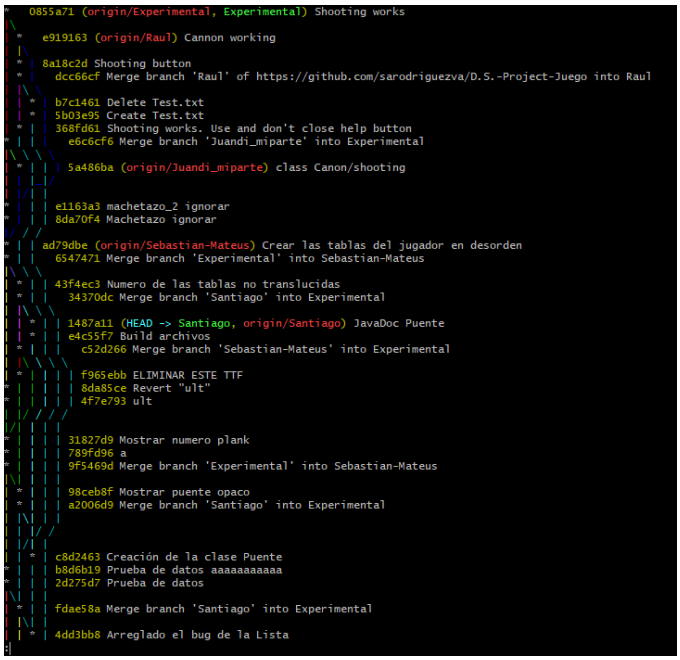
El software se desarrollará en el IDE Netbeans Apache, usando el lenguaje Java junto con la librería LibGDX [6], y funcionalidades de Gradle. A su vez, la gestión del proyecto se realiza mediante un repositorio en Github. [7]

Este software operará en cualquier computador que tenga las siguientes características:

- Java.
- OpenGL.
- Sistema operativo de escritorio.
- Periféricos.
- Al menos 2GB de RAM.

VII. Prototipo de software inicial

El desarrollo del prototipo del programa se hace público en un repositorio en Github creado por los autores [7]. La siguiente es una muestra del trabajo realizado en las ramas del proyecto (generada en Git).



Dicho prototipo incluirá todas las funcionalidades ya mencionadas.

VIII. Implementación y aplicación de las estructuras de datos

El primer nivel del juego utiliza los conocimientos adquiridos en arreglos, listas enlazadas, pilas y colas para poder enseñar al usuario la estructura de las pilas y las colas.

El objetivo del nivel consta que el usuario tendrá que armar un puente con unas tablas que contienen números en un orden determinado, para armar el puente el usuario insertará las tablas (inicialmente dispuestas en orden aleatorio al un costado de la ventana) en un cañón. En cierto punto el usuario tiene la posibilidad de "lanzar" las tablas del cañón al puente y verificará si su respuesta es correcta.

La implementación de las estructuras de datos en este primer nivel se realiza de la siguiente manera:

- Puente:
  El objeto puente no es nada más que un objeto que hereda sus propiedades de una **lista enlazada**. Su

utilidad es definir y verificar el orden en que el usuario deberá acomodar sus tablas (o nodos).
El puente se crea a partir de un **arreglo** que el programa define con anterioridad para el nivel, y contendrá números enteros aleatorios.
Tendrá un método que se encargue de recorrer cada nodo del puente y comparar los elementos con la respuesta dada por el usuario.

- Lista de tablas:
  Las tablas que el usuario podrá mover e insertar al cañón serán objetos tipo "tabla" guardados en una **lista**. Dicha lista solo cumple con el propósito de desplegar visualmente cada tabla en pantalla.

- Cañones:
  Los cañones del primer nivel serán objetos que heredan sus propiedades a partir de una **pila y una cola** respectivamente. El modo en que las tablas se pueden insertar al cañón y cómo estas tablas pueden desplegarse al puente dependerán de la estructura que defina al objeto. Se diferenciarán principalmente por su color y el tutorial del nivel explicará su funcionamiento. Ambos cañones podrán desplegar una visualización de su estado actual como apoyo para el usuario y a su vez se podrán retirar tablas que el usuario defina como insertadas por error.

IX. PRUEBAS DEL PROTOTIPO Y ANÁLISIS COMPARATIVO

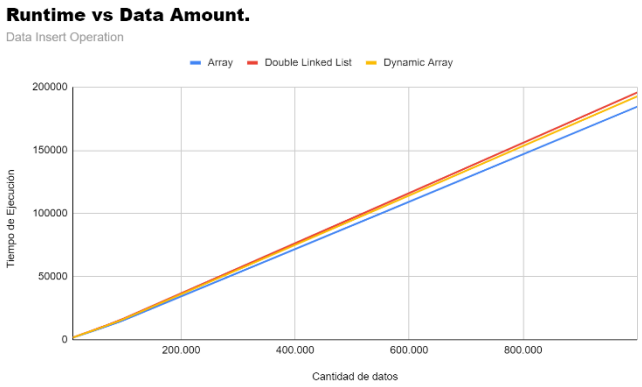
Para determinar el poder computacional de las diferentes estructuras de datos, se determinó una operación de inserción de distintas cantidades de datos, obteniendo los siguientes resultados:

	10.000	
Parámetro	**Runtime (ms)**	**Memory (MB)**
Arrays	1656	35
Double Linked List	1812	35
Dynamic Array	1736	35

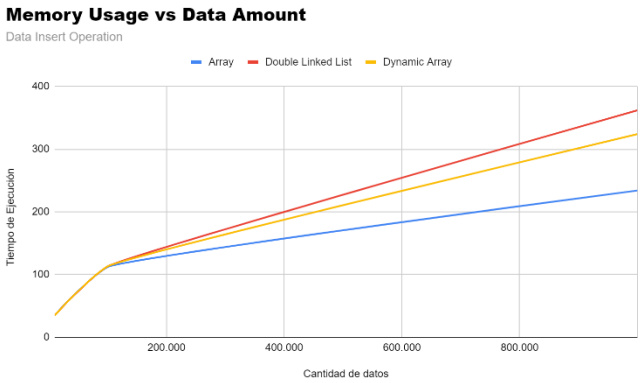
	100.000	
Parámetro	**Runtime (ms)**	**Memory (MB)**
Arrays	15644	112
Double Linked List	16954	113
Dynamic Array	16479	113

	1.000.000	
Parámetro	**Runtime (ms)**	**Memory (MB)**
Arrays	184567	234
Double Linked List	195782	362
Dynamic Array	192770	324

Al graficar estos resultados, se obtiene:



Con esto, se determina que la complejidad completa del programa es O(N), pero la simple operación de inserción para cada estructura es O(1), incluso la del arreglo dinámico porque está amortizada. Se puede apreciar que, con grandes cantidades de datos, la operación de inserción es más útil con los arreglos y se intuye que es debido a la cantidad de instrucciones por operación en conjunto a la gestión de la memoria.



En cuanto al uso en memoria, se puede ver que para pocos datos se suele dar un uso poco efectivo a la memoria, pero a medida que se ingresan más datos, este uso se estabiliza. Se determina que los arreglos gestionan mucho mejor el uso de memoria para grandes cantidades de datos. Le siguen los arreglos dinámicos y entre los 3, una lista doblemente enlazada es la peor opción.

X. ROLES Y ACTIVIDADES

La definición de roles para el presente proyecto se encuentra en la siguiente tabla:

ROL	Actividades fundamentales
Líder(esa)	Consultar a los otros miembros del equipo, atento que la información sea constante para todos. Aportar con la organización y plan de trabajo.
Coordinador(a)	Mantener el contacto entre todos, Programar y agendar y reuniones; ser facilitador para el acceso a los recursos.
Experto(a)	Líder técnico que propende por coordinar las func y actividades operativas.
Investigador(a)	Consultar otras fuentes. Propender por resolver inquietudes comunes para todo el equipo.
Observador(a)	Siempre está atento en el desarrollo del proyecto y aporta en el momento apropiado cuando se requiera apoyo adicional por parte del equipo.
Animador(a)	Energía positiva, motivador en el grupo.
Secretario(a)	Se convierte en un facilitador de la comunicación en el grupo. Documenta (actas) de los acuerdos/compromisos realizados en las reuniones del equipo.
Técnico(a)	Aporta técnicamente en el desarrollo del proyecto.
Programador principal	Implementa el núcleo de las ideas y direccionará el código en general, velando por la facilidad y eficiencia
Programador de apoyo	Se encarga de la creación de código basado en las indicaciones o consejos del programador principal
Tester	Se encargará de probar la funcionalidad, informar sobre los errores que encuentre y ayudará a arreglarlos
Diseñador	Se encarga de crear los recursos gráficos y sonoros además de manejar la experiencia de usuario proporcionando la retroalimentación necesaria para que el programa sea intuitivo

La asignación de los anteriores roles para los integrantes del grupo de trabajo se encuentra a continuación:

INTEGRANTE	ROLES
Juan Sebastian Cabezas Mateus	Lider
	Experto
	Investigador
	Programador principal
	Apoyo en diseño gráfico
Juan Diego Ramírez Lemus	Animador
	Observador
	Técnico
	Programador de apoyo general
	Tester
Raúl Mauricio Peña Losada	Observador
	Técnico
	Programador de apoyo general
	Tester
Santiago Rodriguez Vallejo	Coordinador
	Observador
	Secretario
	Técnico
	Diseñador gráfico principal

XI. DIFICULTADES Y LECCIONES APRENDIDAS

- Gradle no funciona para java 16. Ese error costó tiempo.

- La coordinación para manejar Git no fue la mejor. Es importante que, en las primeras instancias de formulación de proyecto, todos los integrantes tengan un adecuado manejo de esta tecnología.

- El uso de la librería LibGDX tiene una alta complejidad para desarrollar un buen proyecto en poco tiempo, la adaptación a dicha librería no es sencilla y se vuelve algo tedioso cuando distintas personas implementan mismas funcionalidades en formas muy distintas. Pudo haber sido más efectivo usar un motor gráfico, pero gracias a esto se adquirió una colaboración más fuerte en equipo.

- Diseñar el videojuego es el paso más complicado para el desarrollo del proyecto; lo mejor habría sido tener una serie de bocetos previos en dónde se detallen completamente atributos, operaciones e interacciones para agilizar mucho más el trabajo. Esto generó tardanzas e incertidumbre en la asignación de tareas.

- Es complicado demostrar el poder computacional de las estructuras de datos con un proyecto de este tipo, esto en cuanto a poder analizar y comparar los tiempos de ejecuciones de ciertas operaciones; todo esto se debe principalmente a que no hay una entrada directa ni indefinida de datos y la cantidad existente es más bien controlada. Dicha dificultad se soluciona al implementar pruebas experimentales por detrás, pero son cosas que el usuario final no podrá ver ni modificar.

XII. REFERENCIAS BIBLIOGRÁFICAS

[1] A. Widitiarsa, A. W. Utoyo, and M. Sn, "Video Games as Tools for Education," *Journal of Game, Game Art and Gamification*, vol. 03, no. 02, 2018, doi: 10.5281/zenodo.2669725.

[2] D. Dicheva and A. Hodge, "Active learning through game play in a data structures course," in *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, Feb. 2018, vol. 2018-January, pp. 834–839, doi: 10.1145/3159450.3159605.

[3] R. Ramle, D. I. Rosli, S. S. Nathan, and M. Berahim, "Digital game based learning of stack data structure using question prompts," *International Journal of Interactive Mobile Technologies*, vol. 13, no. 7, pp. 90–102, 2019, doi: 10.3991/ijim.v13i07.10778.

[4]"Chapter 8 -- User Input in Games", Dragonlt.atspace.com. [Online]. Available: http://dragonlt.atspace.com/ch8.htm. [Accessed: 25- Apr- 2021].

[5]S. Rodríguez Vallejo, "Mockup VideojuegoDS: Lucidchart", Lucid.app, 2021. [Online]. Available: https://lucid.app/documents/view/bd87e3ba-9e8b-47e4-a1f3-c395036fcfa4. [Accessed: 25- Apr- 2021].

[6]"libgdx/libgdx", GitHub, 2021. [Online]. Available: https://github.com/libgdx/libgdx/wiki. [Accessed: 25- Apr- 2021].

[7]J. Cabezas Mateus, R. Peña Losada, J. Ramírez Lemus and S. Rodríguez Vallejo, "sarodriguezva/D.S.-Project-Juego", GitHub, 2021. [Online]. Available: https://github.com/sarodriguezva/D.S.-Project-Juego. [Accessed: 25- Apr- 2021].