

**Trabajo algoritmos**

Carlos Tarazona Hurtado U201712780

Fabrizio Castellano Castillo U201720728

Aldair Cuarez Baldeon U201521752

1. **Introducción:**

* Problema: El problema consiste en la construcción de un juego infinito con características particulares de los videos juego. Asimismo, considerar que el juego tiene que poseer algún tipo de lógica y deberá mantener coherencia al momento de ser planificado, diseñado e implementado. Por ello, consideramos que nuestro mayor problema al crear este programa será la característica de que el personaje pueda regresar constantemente a cualquier punto del mapa infinito ya recorrido, ya que, deberemos hallar una lógica que pueda hacer esta implementación relevante a el diseño de nuestro juego.
* Objetivo: Los objetivos planteados en la realización de este proyecto serán los siguientes. Ser capaces de implementar satisfactoriamente estructuras de ordenamiento tales como, listas y hashtables, en una interfaz gráfica y haciendo uso de ellas, poder ordenar y distribuir elementos gráficos pertenecientes a los videojuegos. Lograr a su vez, que estas estructuras de ordenamiento nos faciliten una búsqueda rápida y eficiente de las posiciones de algunos de nuestros elementos para así poder detectar algún posible indeseado como un error de colisión o algún enemigo mal posicionado.

1. **Marco Conceptual:**

Nuestro proyecto consiste en emplear datos abstractos y algoritmos de forma adecuada que cumpla con el desarrollo de la competencia del curso. Por lo tanto, es importante tener en cuanto el concepto de los temas que vamos a emplear, Los cuales son:

• Listas Simples: Una lista es un conjunto de elementos llamados nodos que están conectados mediante un puntero. De esta manera, se obtiene una especie de cadena en la cual un elemento apunta al siguiente y así de manera sucesiva. Por esta razón, es una estructura de datos muy utilizada en el ámbito de la programación por la facilidad que brinda al poder insertar valores en cualquier posición de la lista de manera dinámica.

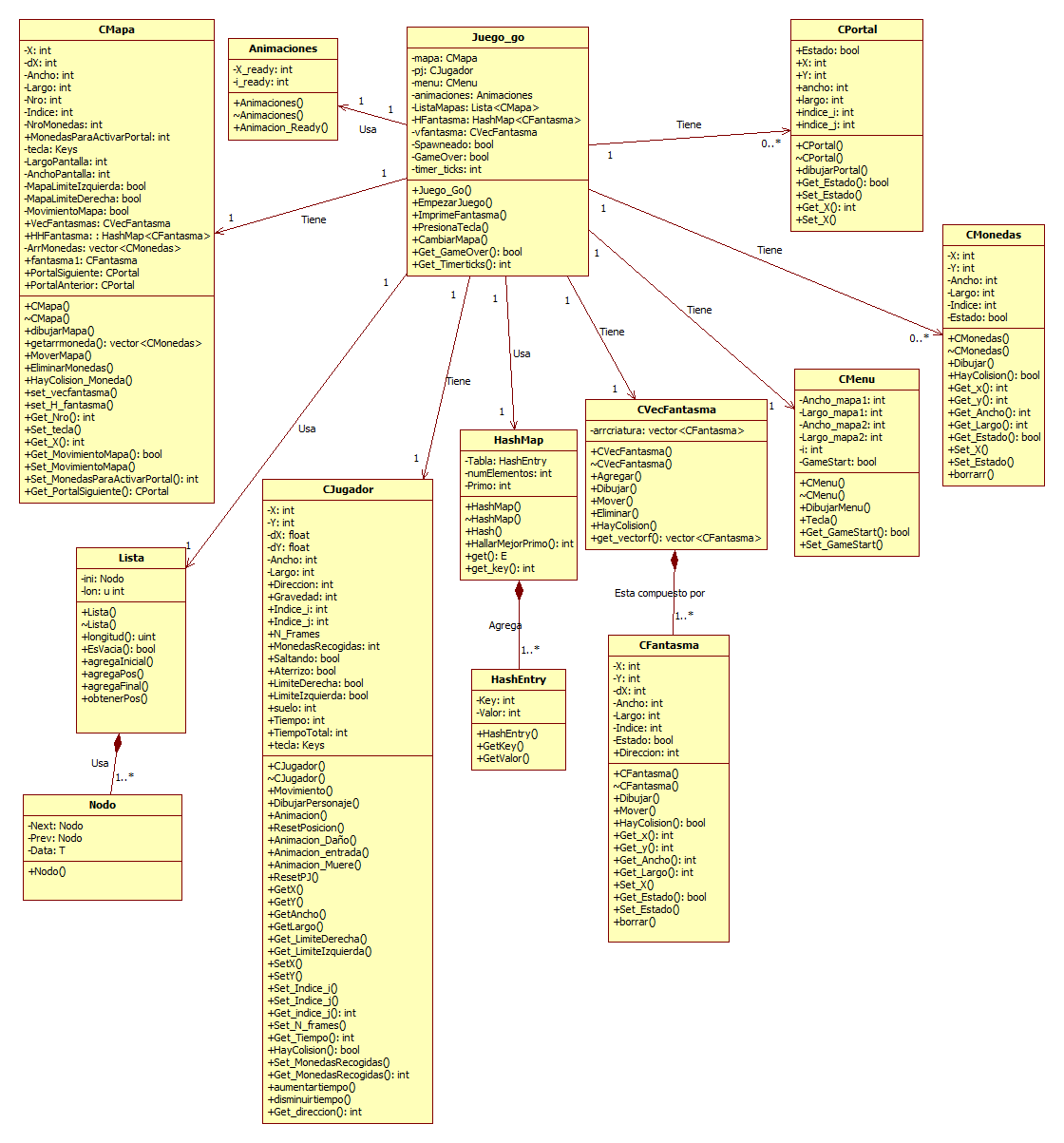
• Las tablas Hash: Es una estructura de datos que puede almacenar grandes cantidades de información, manteniendo las operaciones de inserción, búsqueda y eliminación de manera eficiente y están conformadas por llaves, las cuales apuntan (seleccionan) al lugar donde será almacenada la data, y por los valores, la data que será almacenada en estas tablas. Una de las características de las tablas de hash es la manera en la que se pueden gestionar las colisiones. Debido a que las llaves que se utilizan para insertar su data, es posible para una tabla de hash detectar una colisión de datos cuando más de una de estas llaves se encuentre apuntando a un mismo dato luego de que este fue almacenado, y así

almacenarlo en otro espacio de la tabla donde ninguna llave este apuntando, evitando una colisión.

1. **Requisitos del sistema:**

* Generar el mapa procesualmente por porciones cuando el personaje se desplace por dicho mapa
* Almacenar todas las zonas generadas que se creen
* Generar criaturas por todas las porciones del mapa de manera aleatoria
* La colisión de la criatura y el personaje genera el retorno de dicho personaje a una porción de mapa inferior al que ya estaba
* Generar 1 millón de criaturas bonus
* La colisión de la criatura bonus y el personaje con dos atributos más generan la impresión de otro mapa (otro mundo)
* Las criaturas bonus se generan de manera aleatoria en intervalos diferentes en todo el mapa
* El usuario puede guardar su partida en cualquier momento
* EL usuario puede salir de juego en cualquier momento
* El usuario puede poner pause en cualquier momento

1. **Diagrama de Clases(entidades) UML**



1. **Plan de Trabajo**

Elaboración del trabajo: el trabajo debe estar culminado 3 días previos al día de la entrega de la TF y por lo tanto hemos divido las responsabilidades de los tres miembros en 2 fechas importantes

**Carlos:**

* Implementación del personaje y de la criatura con sus atributos los cuales se puedan desplazar por el mapa de manera adecuada. fecha de culminación (12.11.18).
* Buscar e implementar los Sprites de los personajes, criaturas y los mapas de manera adecuada que se vea de forma amigable y presentable y acorde al juego. fecha de culminación (11.11.18)

**Aldair:**

* Implementación de la función teletransportar en conjunto de la colisión. fecha de culminación (14.11.18)
* La implementación de 1 millon de criaturas por todo el mapa. fecha de culminación (15.11.18)

**Fabrizio:**

* La implementación del guardado del juego en archivos. fecha de culminación (18.11.18)
* Implementación de barras de puntación y de los formularios archivos. fecha de culminación (17.11.18)
* Implementación de los tiempos y validación de que el personaje muera. (17.11.18)

Equipo completo:

* Implementación de un mapa infinito generado procedural mente por porciones las

cuales solo se generan en función a como el jugador va explorando. fecha de culminación (13.11.18)

* Trabajo escrito. fecha de culminación (19.11.18)

1. **Ejecución**
2. **Diseño de interfaz de usuario**

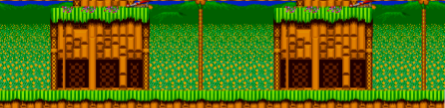
La Interfaz de Usuario(UI) de un [programa](https://www.monografias.com/Computacion/Programacion/) se conforma por elementos hardware y software que muestran [información](https://www.monografias.com/trabajos7/sisinf/sisinf.shtml) al usuario suficiente como para que este entienda lo que debe hacer para proceder con el sistema y lograr que desee con él.



Diseño de la pantalla principal y donde se inicia todo nuestro programa. En esta pantalla podemos encontrar el botón “GAME START” y el botón de “CONTINUE” situados en la parte inferior de la pantalla con los cuales se progresará al principio de los mapas, o en caso de haber guardado la partida, a otro de estos. En la pantalla superior se encentra el logo de nuestro juego, originario del juego Sonic X y un espacio exterior de fondo para que resalte mejor la imagen.



Diseño de mapa A, un mapa propio de los juegos de Sonic, donde tenemos montañas que forman cataratas de fondo y un pasto para que el personaje se mueva.



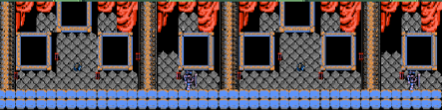
Diseño de mapa B, otro mapa proveniente de los juegos de Sonic, en este podemos encontrar un campo de flores como fondo y el pasto icónico de los primeros niveles de Sonic.



Diseño de mapa C, un tercer mapa de la saga de Sonic con un fondo de desierto, acompañado de unas palmeras y un pasto(piso) por donde se dará el movimiento del personaje.



Diseño de mapa D, un mapa tomado del juego “Mega Man” con un diseño futurista y medio robótico donde podemos observar la iluminación de la ciudad de fondo.



Diseño de mapa E, mapa originario de la saga de juegos “Castlevania” donde podemos observar un castillo decorado de una temática medieval.



Diseño de uno de los mapas con el juego en funcionamiento, se muestra el nivel en el que se encuentra el jugador, el tiempo que le queda para proceder al siguiente nivel antes de que muera, la cantidad de anillos que ha conseguido, el fantasma o enemigo que debe esquivar y finalmente el portal por el que debe pasar para proceder al siguiente nivel, este portal cuenta con un numero en la parte de arriba que indica la cantidad de anillos que se deben conseguir para que se active.

1. **Diseño de interacción**

Es el campo dentro de la experiencia del usuario al usar el sistema, así como la interacción que se genera entre ellos cuando el usuario realiza una acción dentro de la misma. Por ello, realizar un juego como proyecto requiere de varios comandos necesarios para el funcionamiento del mismo.

**Movimientos:**

Al Iniciar el juego el menú se desplegará a través del formulario y te ofrecerá dos opciones

* GAME START
* CONTINUE

Con las que podrás empezar a disfrutar del juego no antes de elegir la opción con las teclas UP y DOWN las cuales se resaltaran al estar encima de una de las opciones para guiarte en tu elección. asimismo, de pulsar la tecla enter para emplear la función y el desarrollo de la misma.



**Movimientos Durante el Juego:**

* Tecla Rigth: Durante el juego al presionar esta tecla podrás desplazar a tu personaje al lado derecho del mapa.



* Tecla Left: Durante el juego al presionar esta tecla podrás desplazar a tu personaje al lado izquierdo del mapa.



* Tecla UP: Durante el juego al presionar esta tecla podrás desplazar a tu personaje en forma vertical por una cantidad limitada de segundos.
* Tecla UP y (izquierda, derecha): Durante el juego al presionar estas teclas juntas tu personaje podrá realizar un salto parabólico por una cantidad de tiempo limitada a la dirección que apunte



1. **Tipos de datos abstractos**

Los Tipos de Datos Abstractos, o TDA son aquellos datos definidos por el mismo desarrollador, y pueden ser controlados con la misma libertad que los tipos de datos ya establecidos por el entorno de desarrollo en el que se esté trabajando. Tiene una estructura basada en valores que son capaces de realizar operaciones básicas entre ellos acorde a su abstracción. Ya que los datos estén adecuadamente establecidos, será posible su declaración y operación con ellas de acuerdo a su establecimiento.

Los TDA utilizados en nuestro proyecto son los siguientes:

Utilizamos una Lista de tipo “CMapa\*” mediante el cual almacenamos los datos referentes a un nivel (información del mapa, vector de monedas, enemigos, etc). Las funciones Implementadas en nuestra lista son “agregaInicial” que inserta un CMapa en un nodo para posteriormente insertarlo al inicio de la lista, “agregaPos” que agrega el nodo en la posición deseada, “agregaFinal” que agrega el nodo al final de la lista y “obtenerPos” que nos retorna el elemento contenido en el nodo, en este caso la Clase CMapa de dicho nodo.

Tambien se utilizó un HashMap de tipo “CFantasma\*” por el cual creamos y distribuimos los enemigos a lo largo de la Lista de tipo “CMapa\*”. Las funciones implementadas en esta estructura son “Hash” que se encarga de almacenar los enemigos en el HashMap y asignarles una llave irrepetible, “HallaMejorPrimo” que su nombre explica por si solo su función, “get” que retorna el enemigo contenido en la posición basándose en el Key del parámetro. Y “Get\_key” que retorna el Key de dicha Posición.

1. **Componentes**

* Controlador:

Es el componente más importante del programa, ya que, en este se realizan todas las acciones para que el juego funcione. Por ello, contiene la mayoría de funciones que cumplen con los requisitos del programa. Por ejemplo: La clase “Juego\_Go” tiene las funciones de cambiar y añadir mapa, funciones primordiales para que se cumpla el juego.

* Estructuras de datos:

El componente que se dedica a almacenar y ordenar los datos para poder cumplir con el guardado de gran cantidad de enemigos y la creación de los mismos. Un ejemplo seria la utilización de las tablas de hash para guardar a los enemigos y el uso de las “key” para crearlos en lugares aleatorios.

* Mapas:

Es el componente que se encarga de almacenar ciertas variables importantes en su class y es fundamental en nuestro desarrollo del movimiento del mapa y la creación de las monedas.

* Animaciones:

Componente de nuestro trabajo que se encarga de realizar las animaciones que funcionan como efectos amigables que interactúa con el usuario para tener una experiencia más satisfactoria con el juego. entre estos tenemos:

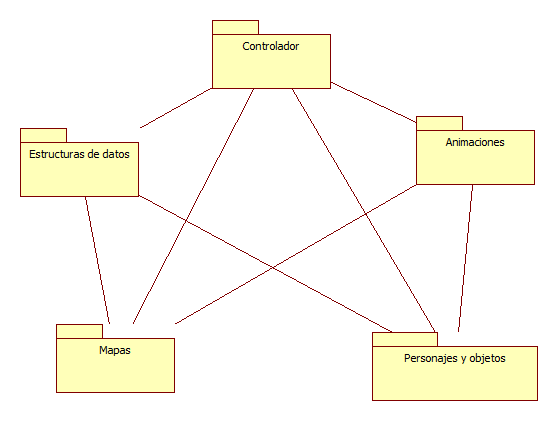
-Animacion \_Ready

-Animacion\_Daño

* Personajes y objetos:

El componente encargado de la mayoría de objetos que se muestran en pantalla y con los que el jugador interactúa, en este componente se encuentran los tamaños de los enemigos y del personaje, su posición, estado (si está vivo o muerto), y otros.

1. **Diagrama de Clases(componentes) UML**

Representa como un software o programa es dividido en diversas componentes y muestra la relación que tienen entre ellas.

1. **Conclusión**

A través de este trabajo hemos concluido que desarrollar un videojuego aplicando Tipos de Datos Abstractos nos permite mayor libertad creativa ya que nos brindas más oportunidades que usando los conceptos básicos de Programación Orientada a Objetos. Además, La Implementación de diferentes estructuras de datos nos concedió una visión más panorámica y concisa de como deber realizarse un trabajo.

1. **Referencias**

- Charte Ojeda, Francisco (Fecha de consulta: 10 de Noviembre de 2018). “Programación con C++ Builder 2006”. fcharte.com (<http://www.fcharte.com/libros/84-415-1988-9.pdf>).

- Garrido Carrillo, Antonio (Fecha de consulta: 11 de Noviembre de 2018). “Abstracción y Estructuras de Datos en C++”. Granada, España (<https://books.google.com.pe/books?hl=es&lr=&id=qXoVzD23DBsC&oi=fnd&pg=PA2&dq=c%2B%2B+listas&ots=7V0CCZeQS1&sig=Fl5NpI7PpNbCA7BpKE3BUi23lcw#v=onepage&q=c%2B%2B%20listas&f=false>).

- Hundt, Robert (Fecha de consulta: 11 de Noviembre de 2018). “Loop Recognition in C++/Java/Go/Scala”. scala-lang.org (<https://days2011.scala-lang.org/sites/days2011/files/ws3-1-Hundt.pdf>).