

***HITO 3***

**Asignatura:**

Algoritmos y estructura de datos

**Ciclo:**

2018-2

**Integrantes:**

* Maximo Ali Mallqui Tertuliano
* Fernando Rey Arévalo Díaz
* Ricardo Raúl More Molina

**Profesor:**

Luis Martín Canaval Sánchez

LIMA - PERÚ

2018

**Super Lag - Mingyu y sus amigas**

**Introducción**

Con el fin de desarrollar la construcción de un juego tipo **side-scroller** y desarrollar la mecánica principal del juego la cual consiste en un retardo de tiempo de respuesta al realizar una acción del personaje, ante esto proponemos una solución innovadora mediante la utilización de colas, esto nos ayudará a capturar la acción del personaje para luego después de un tiempo determinado, desencolarlo para luego ejecutarlo. Además de implementar una plataforma multijugador.

**Objetivos**

* Implementar un juego de tipo **side-scroller**, juego en el cual el personaje se desplaza de manera lateral.
* Implementar la mecánica principal de **Retardo**, la cual consiste en que toda acción que realice el personaje se vea reflejada en el juego después de unos segundos.
* Implementar la plataforma multijugador.
* Implementar obstáculos que los personajes deberán esquivar.
* Implementar enemigos los cuales los personajes deberán evitar pues les reducen la vida

**Marco Teórico**

La temática principal del juego se basa en que los personajes principales Mingyu, Mely y Ponky las cuales deben evadir a sus enemigos o eliminarlos. Además, los jugadores deben ser muy cuidadoso para no impactar contra el enemigo, puesto que poseen el **Retardo** al moverse, lo cual hace que el juego tenga mayor dificultad. Para lograr esto, usaremos árboles AVL y colas.

ARBOLES AVL :

Un árbol AVL es un tipo especial de árbol binario ideado por los matemáticos rusos Adelson-Velskii y Landis (A-V-L). Fue el primer árbol de búsqueda binario auto-balanceable ideado.

Los árboles AVL están siempre equilibrados de tal modo que para todos los nodos, la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa. Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en orden de [complejidad](https://es.wikipedia.org/wiki/Complejidad_computacional) [O](https://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica)(log n). El factor de equilibrio puede ser almacenado directamente en cada nodo o ser computado a partir de las alturas de los subárboles.

Cada nodo, además de la información que se pretende almacenar, debe tener los dos punteros a los árboles derecho e izquierdo, igual que los [árboles binarios de búsqueda](https://es.wikipedia.org/wiki/%C3%81rbol_binario_de_b%C3%BAsqueda) (ABB), y además el dato que controla el factor de equilibrio.

El factor de equilibrio es la diferencia entre las alturas del árbol derecho y el izquierdo:

FE = altura subárbol derecho - altura subárbol izquierdo

Por definición, para un árbol AVL, este valor debe ser -1, 0 o 1.

Si el factor de equilibrio (FE) de un nodo es:

0 -> el nodo está equilibrado y sus subárboles tienen exactamente la misma altura.

1 -> el nodo está equilibrado y su subárbol derecho es un nivel más alto.

-1 -> el nodo está equilibrado y su subárbol izquierdo es un nivel más alto.

Si el factor de equilibrio es igual o mayor a 2,{\displaystyle |Fe|>=2} es necesario reequilibrar.

El reequilibrado se produce de abajo hacia arriba sobre los nodos en los que se produce el desequilibrio. Pueden darse dos casos: rotación simple o rotación doble; a su vez ambos casos pueden ser hacia la derecha o hacia la izquierda.

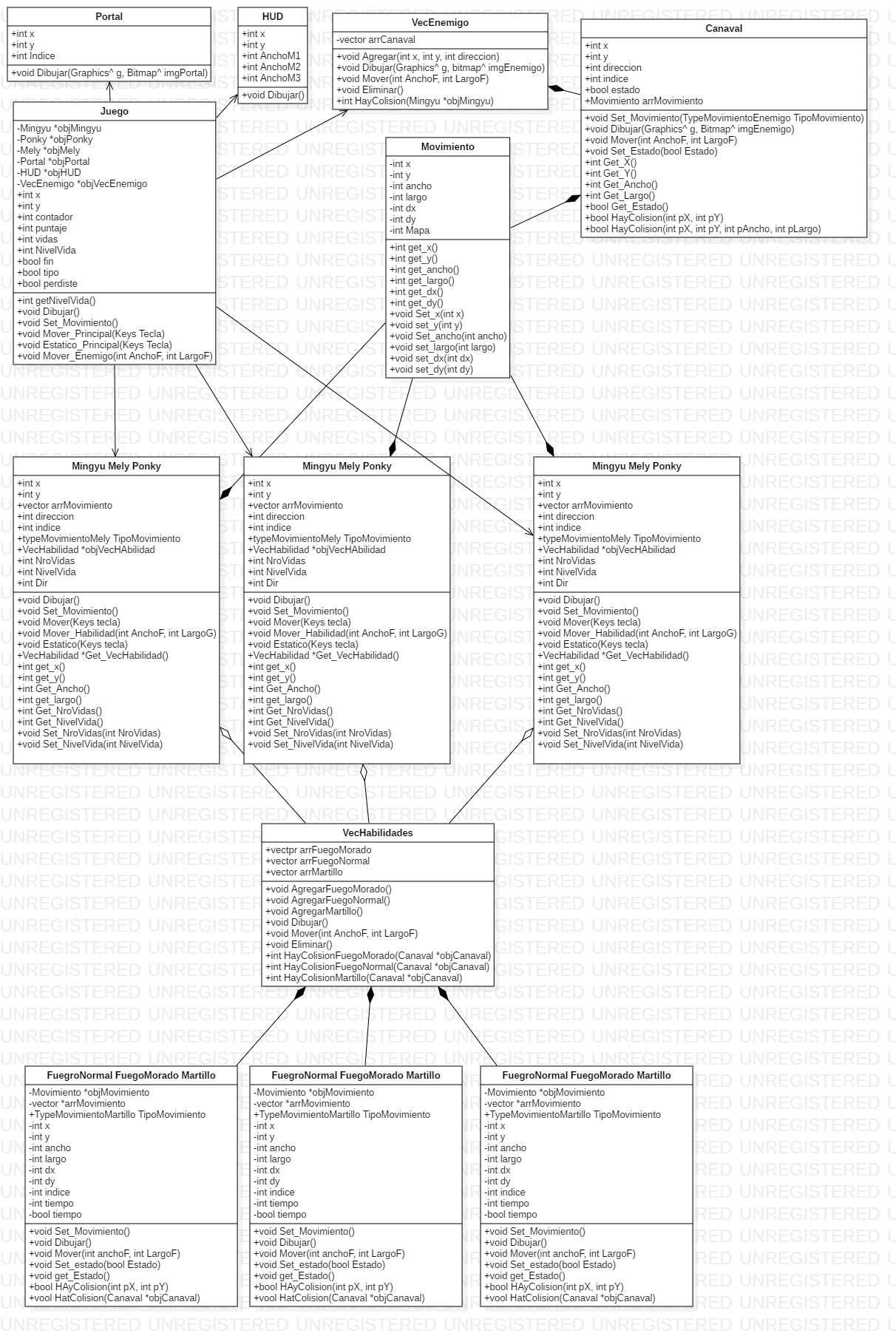
COLAS:

Son estructuras utilizada como herramientas de programación del tipo FIFO (First in -First out). El cual simula una cola de cine, de supermercado y/u otros, aunque esta es una colección ordenada de objetos.  
Básicamente poseen tres operaciones primarias:   
Encolar dato: inserta el dato al final de la cola.  
Desencolar dato: remueve e dato al inicio de la cola.  
Peek : permite visualizar el dato al inicio de la cola.

REQUISITOS

-Programación orientada a objetos  
-Colas

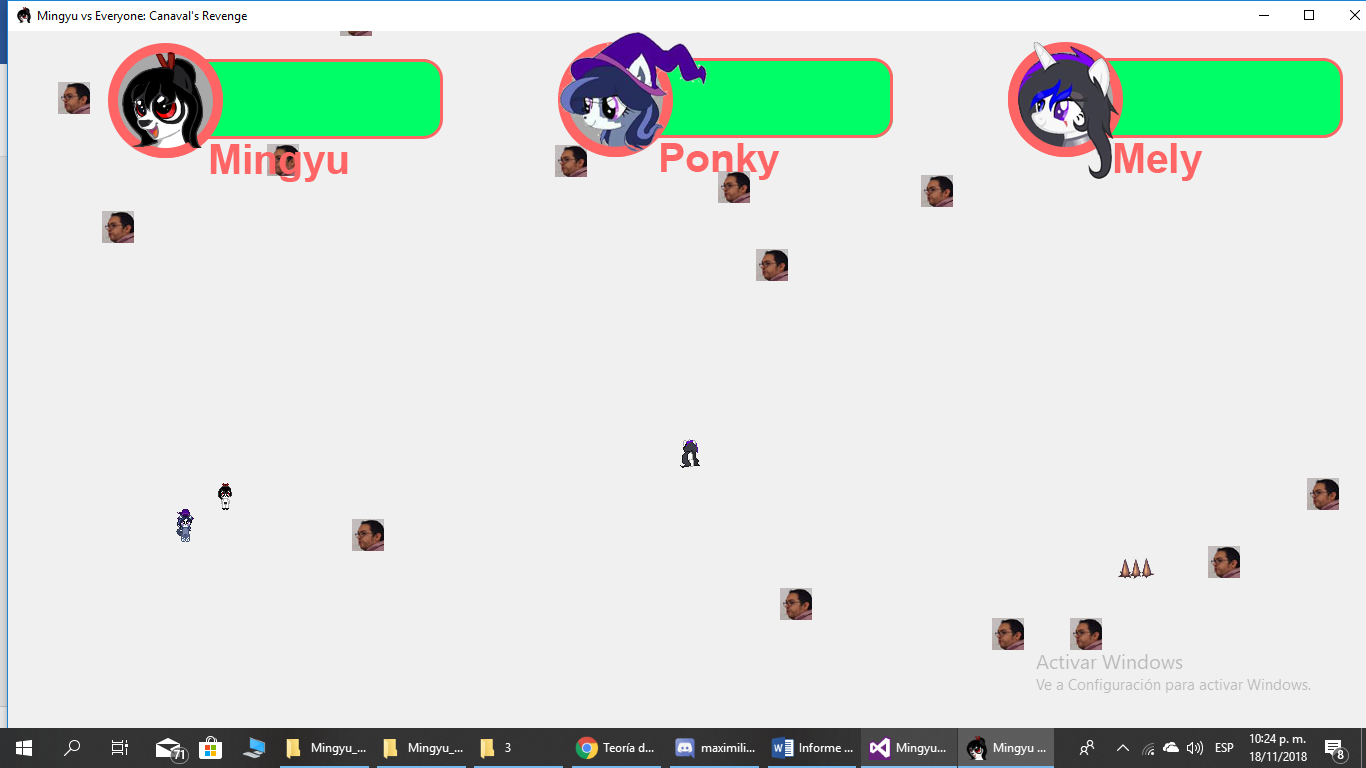
**Diagrama de Clases**



|  |  |  |
| --- | --- | --- |
| Cronograma de trabajos | | |
| **Actividad** | **Plazo** | **Encargado** |
| Manejo de árboles AVL | 19/10/2018 - 21/11/2018 | Fernando Arévalo |
| Manejo de historia | 19/10/2018 - 21/11/2018 | Maximo Mallqui |
| Patrones de enemigos | 11/11/2018 - 21/11/2018 | Ricardo More |
| Funciones de los personajes jugables | 17/10/2018 - 21/11/2018 | Maximo Mallqui |
| Control de calidad | 23/10/2018 - 28/11/2018 | Ricardo More |
| Inspección de bugs | 13/11/2018 - 28/11/2018 | Fernando Arévalo |
| Funcionalidad de mapas | 17/11/2018 - 24/11/2018 | Maximo Mallqui |
| Guardado de partida | 17/11/2018 - 24/11/2018 | Fernando Arévalo |
| Asistencia técnica | 14/11/2018 - 28/11/2018 | Ricardo More |

**Plan de trabajo**

**Ejecucion**

1. Diseño de Interfaz de Usuario
2. Diseño de interacción

La interacción entre Usuario y Juego es mediante las teclas del computador y/o joysticks.

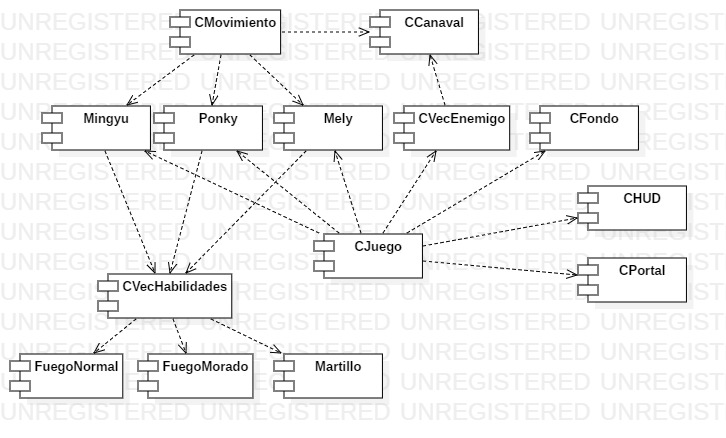
1. Tipos de datos abstractos

* Colas
* ArbolesAVL

1. Componentes

* CJuego
* CHUD
* CPortal
* Cfondo
* CVecEnemigo
* CCanaval
* CMovimiento
* Mingyu
* Ponky
* Mely
* CVecHabilidades
* FuegoNormal
* FuegoMorado
* Martillo

1. Diagrama de clases (componentes)



**Conclusiones**

* Los Árboles AVL son de gran utilidad para realizar búsquedas en grandes cantidades de datos.
* Realizar la búsqueda en un millón de Mapas por un atributo se hace mas simple con arbolesAVL.
* La Gestión de memoria es importante ya que de no ser así se produce un StackOverFlow.
* La Mecánica de retraso se logró implementar gracias a la estructura de dato “Cola”, ya que nos permite almacenar el movimiento del personaje para luego desencolarlo.

**Referencias**

Wikipedia. Arboles AVL . Recuperado de https://es.wikipedia.org/wiki/%C3%81rbol\_AVL