UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Estructura de Datos B

Ing. Carlos Gustavo Alonzo

Aux. Josué David Itzep Salvador

PROYECTO #1

Battle-Cys

La compañía NanoHard, es una empresa líder en desarrollo de tecnología. En la actualidad deben desarrollar un juego para atraer más clientes, de todos los sectores del mercado, por lo que se ha asociado con los estudiantes de Estructura de Datos de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, para desarrollar dicha solución.

Descripción

El juego constara de una arquitectura cliente-servidor, y se conectara mediante Apache Thrift, la parte del servidor será desarrollada en el lenguaje C/C++, y el lado del cliente será desarrollado en Java. El cliente poseerá un tablero que servirá como mando de juego, aquí se ejecutaran los movimientos del jugador, se podrá visualizar el puntaje actual, y poseerá una cantidad mínima de estructuras en comparación con el servidor. En el servidor se podrá realizar la configuración inicial del juego, pudiendo establecer el número de niveles, el tamaño de los niveles, punteos, etc. además se podrá jugar de manera automática mediante el uso de codigos para completar los niveles de una manera más rápida.

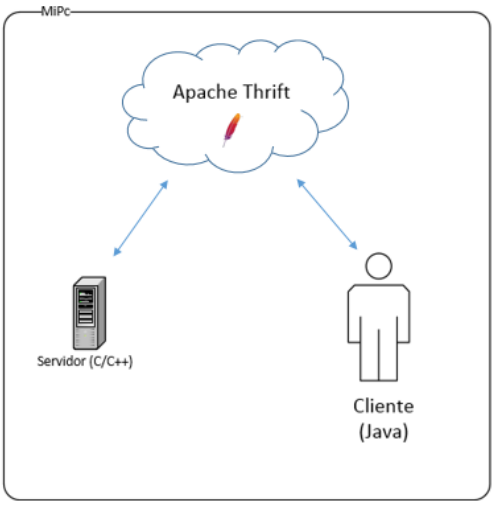

Figura 1. Arquitectura

Servidor

El servidor será desarrollado en el lenguaje C/C++, y deberá de poseer una GUI amigable, entretenida, fluida, e intuitiva, para lo cual se utilizara QtCreator.

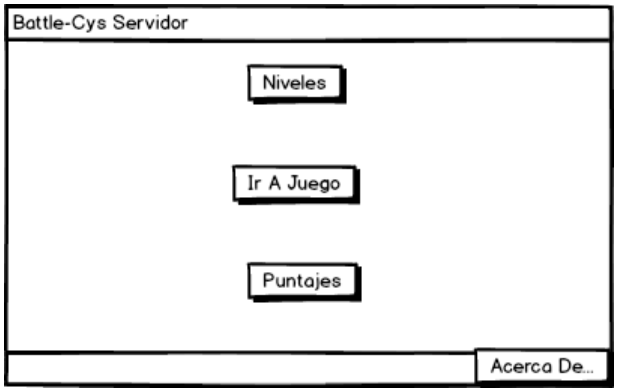

Figura 2. Servidor

NIVELES

El número de niveles N deberá ser ingresado por el administrador del servidor, este podrá ser aumentado o disminuido en cualquier momento del desarrollo del juego. De la misma manera se tendrá una **lista doblemente enlazada** la cual deberá contener un nivel en cada uno de sus nodos, permitiendo así, acceso a todos los niveles. Un nivel consiste en una **matriz ortogonal**, cada una de las matrices contara con sus apuntadores nulos al inicio, ósea no contendrá nodos, ni cabeceras, solo su nodo raíz. Y se podrá visualizar de manera gráfica en cualquier momento y estado.

De la misma manera todos los niveles deberán estar BLOQUEADOS al inicio del juego, a excepción del nivel inicial. Para que el cliente/usuario pueda moverse de nivel deberá de “ganar” el nivel previo para desbloquear el nivel siguiente, y poder seguir avanzado en la lista

de niveles, además se podrá mover en cualquier dirección, ya sea a un nivel mayor o a un nivel menor.

TAMAÑO NIVELES

El tamaño T que tendrán los niveles será ingresado por el administrador del servidor, y será el mismo para las dimensiones (x,y) de cada uno de los niveles (e.g. T=4, nivel/matriz de 4*4), esta podrá ser incrementada o disminuida en cualquier momento del desarrollo del juego, dicha dimensión será la máxima que puede tener cada uno de los niveles. Todos los niveles poseerán la misma cantidad de dimensiones (x,y) y el mismo tamaño T.

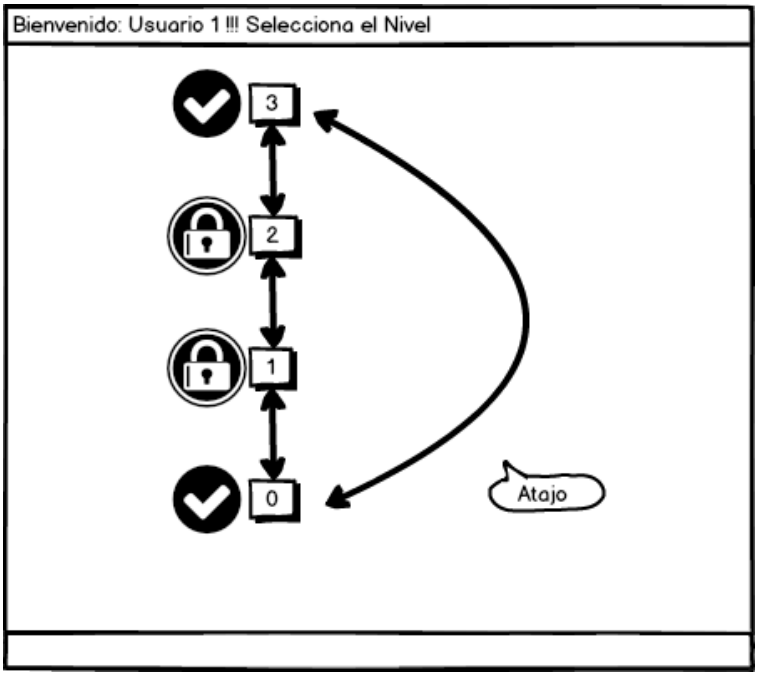

Figura 3. Lista de Niveles

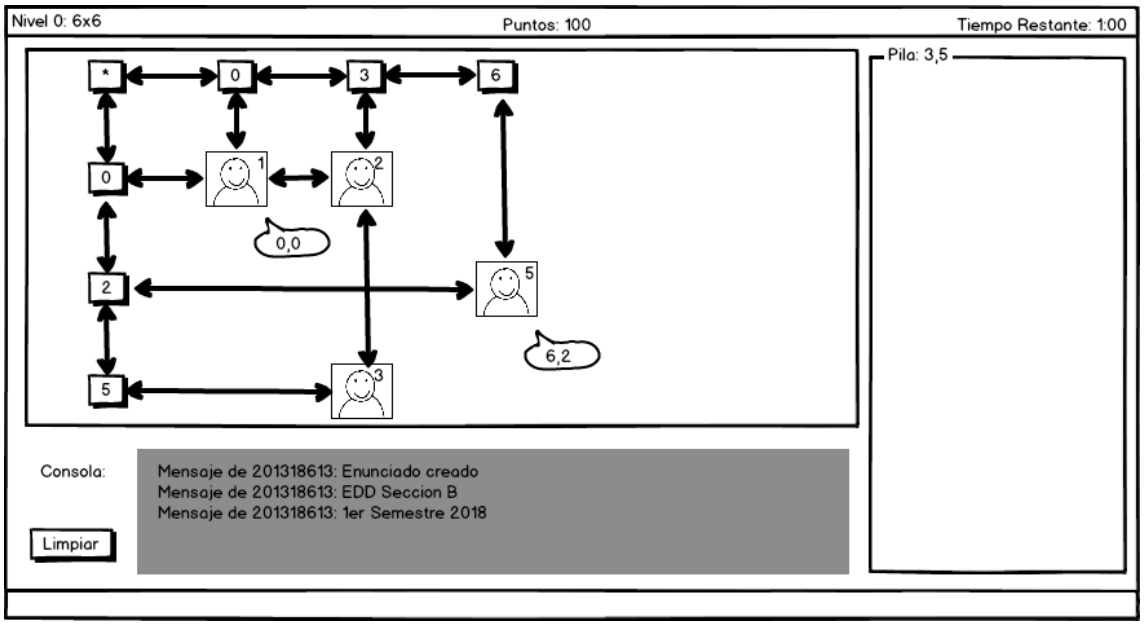

Figura 4. Nivel 'n' de 6x6

- Se debe poder visualizar los niveles como un todo (lista).

PILA DE ENEMIGOS/BONOS

Cada nivel en cada uno de sus nodos (a excepción de las cabeceras), poseerá **pilas** en las cuales se posicionaran ya sea bono (ver sección “Bonos Diamante”) o los enemigos, que se describen a continuación.

ENEMIGOS

Se tendrá 3 tipos de enemigos, cada enemigo tendrá un ID, tendrá distinto aspecto (entre c/tipo de enemigo), el ID debe de ser visible en todo momento. Los tipos de enemigos serán basados en la dificultad que tendrá el cliente para eliminarlos del juego. Los tipos son:

- Nivel Bajo: Serán eliminados con 1 acierto
- Nivel Medio: Serán eliminados con 3 aciertos
- Nivel Alto: Serán eliminados con 5 aciertos

Se crearan enemigos cada 8 segundos, el tipo de enemigo será aleatorio, y se podrá pausar la creación de enemigos en cualquier instante del juego (por cada nivel), esto pausara también el tiempo de sesión y el tiempo transcurrido en el nivel. Por cada enemigo eliminado, se dará un puntaje al usuario/cliente, el puntaje será basado con respecto a la siguiente tabla:

Nivel	Puntos
Bajo	10
Medio	20
Alto	40

*De la misma manera se deberá poder visualizar la pila que posee cada nodo, y los elementos que hay en esa pila, ya sean enemigos o bonos, se realizara mediante el ingreso de una coordenada (x,y)

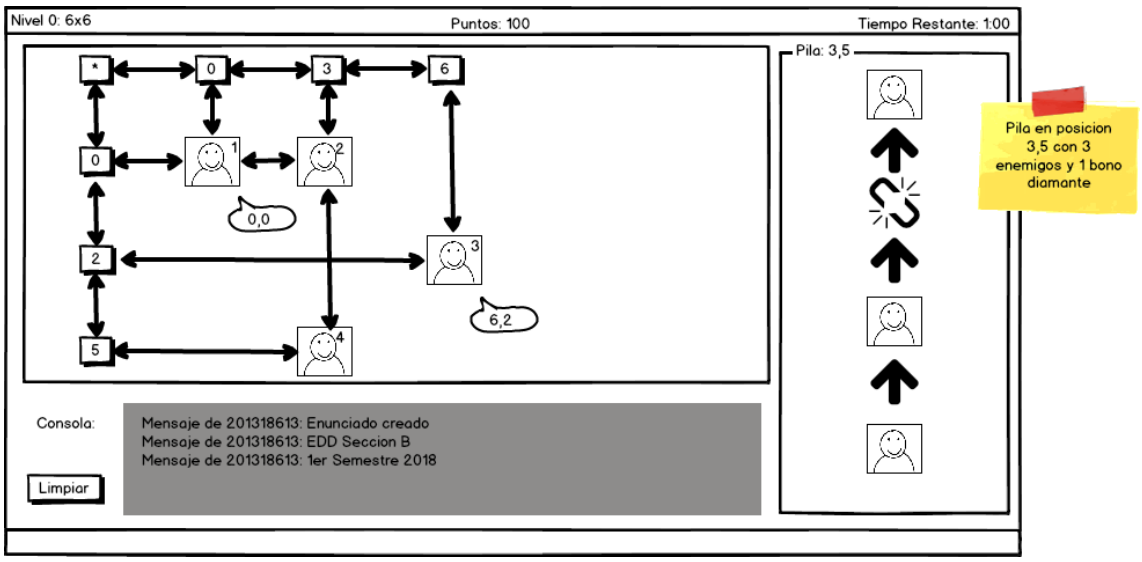

Figura 5. Visibilidad de pila de un nodo

MOVIMIENTO DE ENEMIGOS

Los enemigos deberán de moverse a lo largo del nivel (por cada nivel), de manera automática, lo cual implica una animación, y la realizaran cada 4 segundos. La posición inicial de los enemigos al momento del inicio del juego, posterior a su creación será calculada mediante dos números aleatorios, que oscilaran entre 0,T (que es el tamaño del nivel mencionado en la sección "Tamaño niveles") para la coordenada 'X'; de la misma manera será calculada la posición 'Y'. El movimiento en el tiempo de juego solo se realizara en una unidad para cada coordenada (e.g. posición actual enemigo: 4,4 – nueva posición del enemigo: 5,4 o 5,4 o 4,3 o 3,4 – movimientos inválidos: 3,3 o 5,5 etc.), la dirección en la que realizaran el movimiento también será aleatoria siendo:

- 0 – arriba
- 1 – derecha
- 2 – abajo
- 4 – izquierda

De realizar un movimiento que se salga del rango de la matriz, se calculara una nueva coordenada hasta que realice un movimiento correcto.

Dado a que existirán pilas en cada uno de los nodos del nivel, de encontrarse más de un enemigo en la pila, el único que podrá realizar movimientos deberá ser aquel que se encuentre al tope de la pila. Todos los enemigos que se encuentren en el tope de la pila realizaran movimientos a los 8 segundos.

TIEMPO DE JUEGO

Se deberá mostrar en todo momento el tiempo de juego total en el que el usuario haya estado jugando durante toda la sesión, así como el tiempo que se lleva en cada nivel. Cada nivel tendrá un máximo de tiempo que ira de acuerdo a la siguiente tabla:

Nivel	Tiempo
1-2	1:00
3-4	1:30
5-6	2:00
>7	3:00

*De terminarse el tiempo disponible para el nivel, se dará por perdido el nivel. Se detendrán las animaciones, se enviara un mensaje al cliente que el tiempo ya ha terminado, y se mostraran la lista de niveles, a no ser que el cliente haya alcanzado el puntaje que se muestra en la siguiente sección, de ser así el nivel se dará por ganado el nivel.

FIN DE NIVEL

Si se alcanzan los siguientes puntajes por niveles se dará por ganado el nivel

Nivel	Puntaje
1	80
2	100
3	120
4	140
5	160

Para los niveles superiores se ha aumentaran 20 puntos al puntaje del nivel anterior requeridos para ganar, de la siguiente manera: Nivel n, puntaje: Puntaje(n-1)+20.

Ejemplo:

Nivel 6 -> Puntaje: Puntaje(5)+20 = 160 + 20 = 180

Nivel 7 -> Puntaje: Puntaje(6)+20 = 180 + 20 = 200

*Al terminar el nivel se deberá mostrar el siguiente mensaje

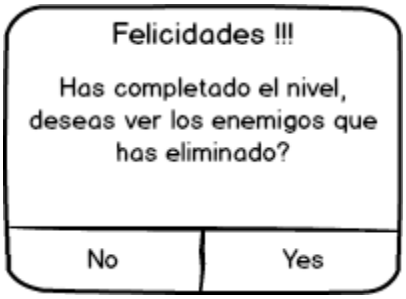

Figura 6. Mensaje fin de nivel

De seleccionar la opción “Si”, los enemigos se mostraran ordenados por id, en una lista circular simplemente enlazada, cada tipo de enemigo poseerá una propia lista como se muestra a continuación.

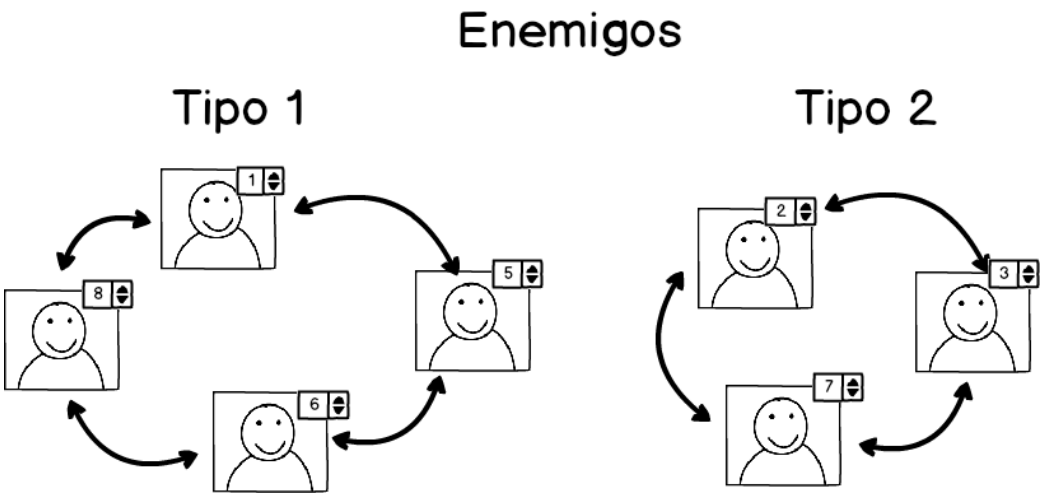

Figura 7. Listas circulares de enemigos.

CONSOLA

Se deberá crear una consola integrada a la GUI, la cual muestre cada uno de los movimientos y ataques realizados por el usuario/cliente, movimientos de los enemigos, posiciones donde se creen enemigos, bonos, enemigos reposicionados, puntajes obtenidos por cada ataque, vida restante de cada enemigo, y demás información necesaria para el control de puntajes y transacciones, esta deberá ser visualizada en cualquier momento, deberá demostrar la información de una manera amena y ordenada (ver Figura 5), así mismo se tendrá un botón para limpiar los mensajes que haya en consola. Cada mensaje deberá llevar el mensaje "Mensaje de #carnet", antepuesto antes de cada movimiento, donde #carnet es el carnet del estudiante.

Ejemplos:

Mensaje de 201318613: Enemigo #ID de tipo #tipo creado en posición x1,y1

Mensaje de 201318613: Bono diamante creado en posición x2,y2

Mensaje de 201318613: Ataque realizado en posición x3,y3

Mensaje de 201318613: Puntos: +10

Mensaje de 201318613: Vidas restantes: #vidas de enemigo #id

Mensaje de 201318613: Ataque fallido en posición x4,y4

Mensaje de 201318613: Movimiento de enemigo #id hacia #direccion a posición x5,y5

Mensaje de 201318613: Movimiento de enemigo #id hacia #direccion invalido

Mensaje de 201318613: Bono diamante en posición x6,y6 ha expirado

Mensaje de 201318613: Se han acumulado 3 bonos diamante tienes derecho a un atajo

Mensaje de 201318613: Error 404 – Enemigo reubicado en posición x7,y7

Mensaje de 201318613: Cheat code ingresado, enemigo en x8,y8 eliminado

BONO DIAMANTE

En el momento que se eliminen 5 enemigos de manera consecutiva, aparecerá en un lugar aleatorio dentro del nivel con coordenadas (x,y), calculadas aleatoriamente, un bono diamante. Si el cliente/usuario, realiza un ataque a una posición donde se encuentre en el tope de la pila uno de estos bonos, el bono desaparecerá, y se acumulara por nivel, por consecuencia la acumulación de estos solamente será válido para un nivel. Los bonos podrán estar en cualquier posición dentro de la pila, esto significa que cuando se cree un bono siempre ira en el tope de la pila, sin embargo sobre él pueden posicionarse o seguir siendo creados enemigos. El bono desaparecerá a los 5 segundos de su creación, por lo que tendrá un tiempo de vida limitado.

ATAJOS

Al momento de acumular 3 bonos diamante dentro de un nivel, se guardara esta bandera y al momento de finalizar el nivel, ya sea porque se termine el tiempo, o porque el usuario gane o pierda el juego, posteriormente este podrá desbloquear un nivel cualquiera dentro de la lista de niveles (superiores), por lo que NO será necesario que haya ganado los niveles anteriores al nivel seleccionado, y se referenciara el nivel actual donde se acumularon los 3 bonos hacia el nuevo nivel, si se posiciona dentro de un nivel dentro de la lista de niveles donde haya obtenido un atajo, este podrá ir al nivel superior o al nivel inferior al que apunta el atajo.

ERROR-404

Cuando un cliente/usuario, se equivoque de objetivo, ósea seleccione un nodo vacío/nulo 5 veces consecutivas, el ultimo enemigo eliminado deberá de reaparecer en las posición donde fue eliminado, esto se repetirá hasta que no hayan enemigos previamente eliminados. De la misma manera el usuario/cliente perderá 100 puntos.

PUNTAJES

Se deberá de visualizar los distintos puntajes de los usuarios que han jugado han obtenido, dichos puntajes podrán visualizarse como un todo a través de una **lista doblemente enlazada modificada** (ver Figura 8 y observaciones), de los cuales se podrá seleccionar cualquiera de estos. De seleccionarse un usuario se mostrara los puntajes obtenidos en cada uno de los niveles. Los puntajes estarán almacenados en un **árbol binario de búsqueda** (ver Figura 9).

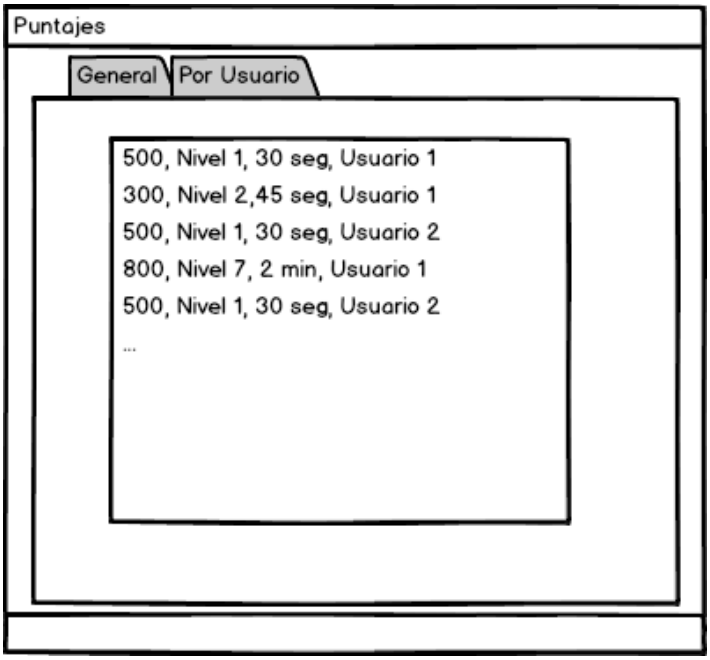

Figura 8. Lista de puntajes, representación lógica, se deberá mostrar representación gráfica indicada en observaciones

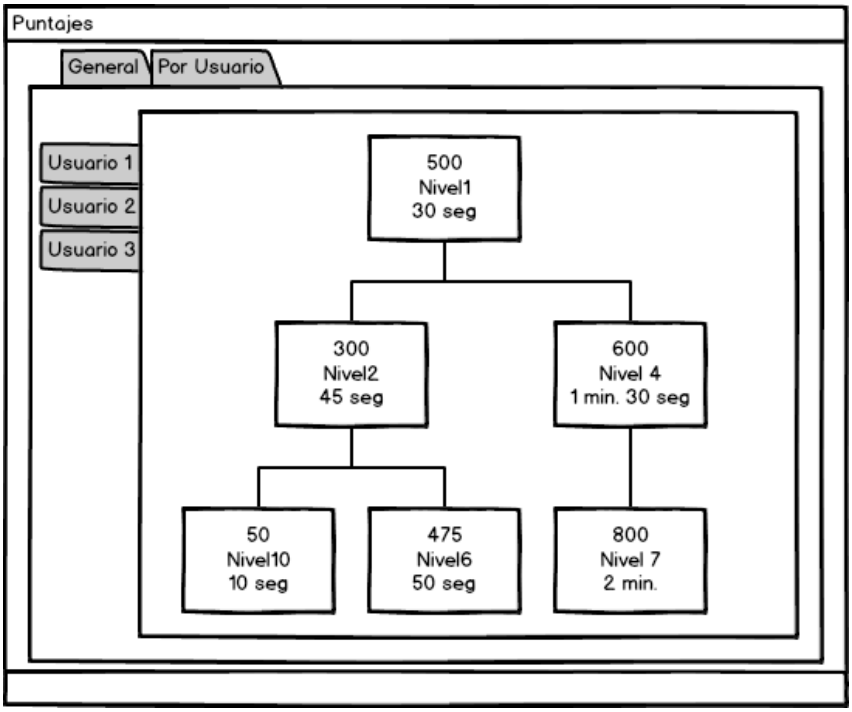

Figura 9. Puntajes por usuario, en ABB

CHEAT CODES

El cliente ingresara códigos, con los que podrá realizar ataques certeros a los enemigos, de este lado de la arquitectura se comprobara que el código ingresado sea correcto. Este eliminara el último enemigo creado, por lo que se tendrá también una **pila** de posiciones de los enemigos. Si el ataque se hace sobre un bono diamante, este solo se eliminara, sin embargo no se otorgara los puntos para el atajo. Se explica más adelante los códigos.

Cliente/Usuario

El cliente será desarrollado en el lenguaje Java. Así como el servidor, este mismo deberá de poseer una GUI amigable, entretenida, fluida, e intuitiva, para lo cual se utilizara el IDE que mejor le parezca al estudiante.

Figura 10. Aplicación Cliente

MANDO DE JUEGO

El mando de juego constara de una **matriz ortogonal estática**, la cual tendrá el mismo tamaño que se ingresó por el administrador del servidor del otro lado de la topología.

PUNTAJE y TIEMPO

El cliente deberá de poder visualizar en tiempo real el puntaje que ha obtenido en el desarrollo de un juego, así como, el tiempo transcurrido por nivel.

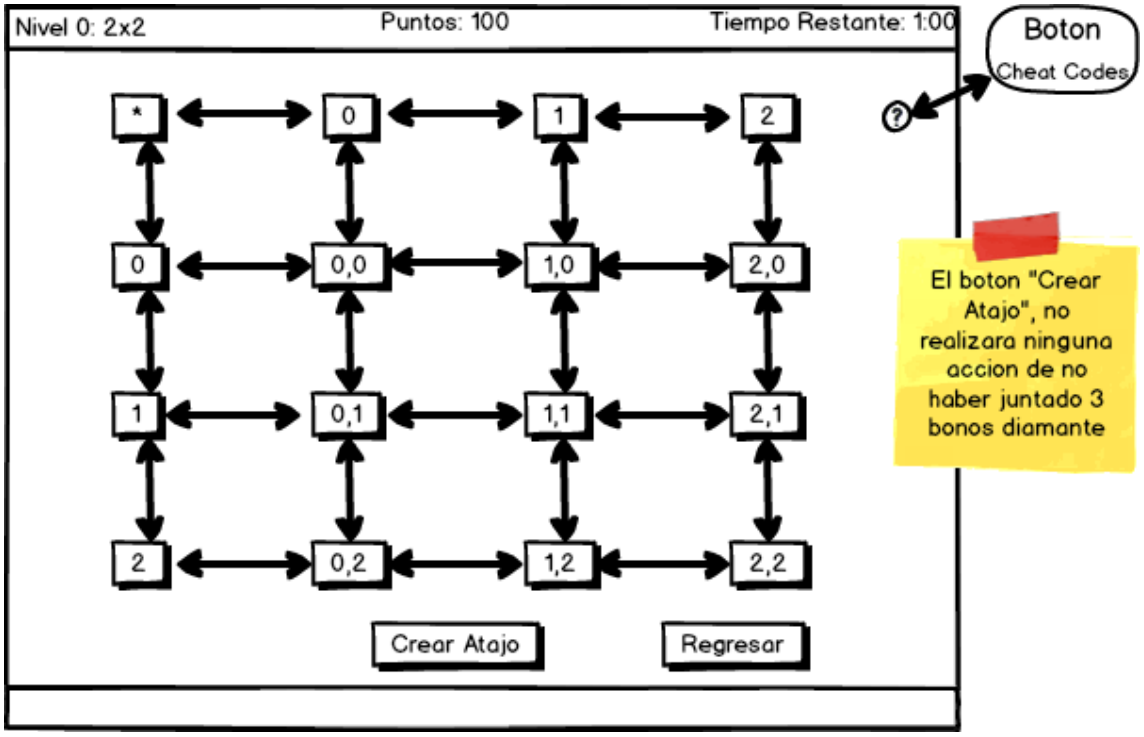

Figura 11. Mando de Juego usuario, matriz ortogonal Estática

CHEAT CODES

Existirá un botón de “cheat codes” (ver Figura 11), el cual mostrara una ventana donde se podrá ingresar el siguiente código y enviarlo al servidor, el cual eliminara el ultimo enemigo creado en el servidor.

“edd-b-practica1-#carnet”

Donde carnet será el #carnet del estudiante. Ejemplo:

“edd-b-practica1-201318613”

Esto incrementara el ritmo de juego, lo que quiere decir que los movimientos de los enemigos, creación de enemigos, y demás acciones mencionadas anteriormente reducirán a la mitad su tiempo de ejecución (a excepción del tiempo de juego), además realizara ataques automáticos durante el tiempo de ejecución de un nivel de juego a cada 5 segundos.

SELECCIÓN NIVEL

Se deberá de poder seleccionar el nivel, esta GUI será solo de movimientos, los niveles serán visualizados únicamente en el servidor y la posición del usuario se deberá de visualizar en

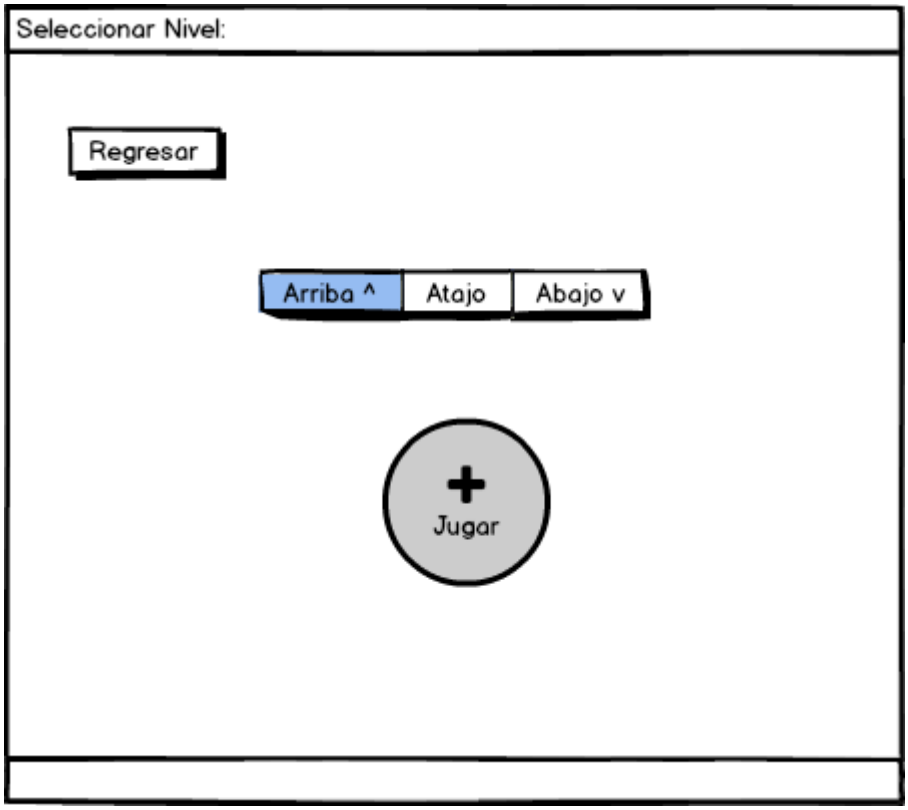

Figura 12.Seleccion Nivel, usuario

GENERAL

Tanto en el servidor como en el cliente existirán botones “Acerca De …”, con los que se desplegara información acerca del curso, semestre, y datos relevantes del estudiante como nombre, carnet, CUI.

Observaciones

- Sistema Operativo: Linux
- IDE's permitidos: QtCreator, Netbeans
- Herramienta de Comunicación: Apache Thrift
- Las estructuras deben de ser realizadas por el estudiante, deben usar struct y punteros C/C++ para el servidor y clases para el cliente Java
- Herramienta de Graficas: Graphviz
- Las GUI mostradas en este enunciado, son para orientar al estudiante sobre lo requerido, el estudiante debe de realizarlo como mejor le parezca
- Implementación para carnet Impar:
    - Matriz ortogonal – punteros hacia izquierda y hacia abajo
    - Lista puntajes: Nodo cabeza será el último registro indicado, punteros anteriores

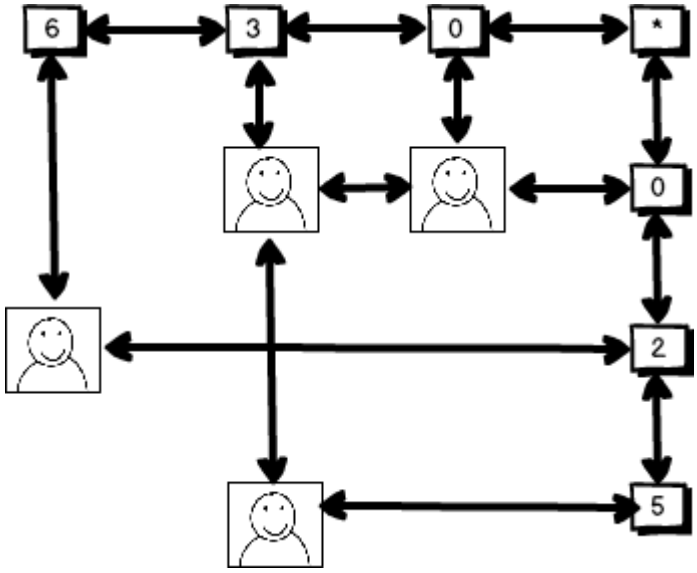

Figura 13. Matriz ortogonal imprares

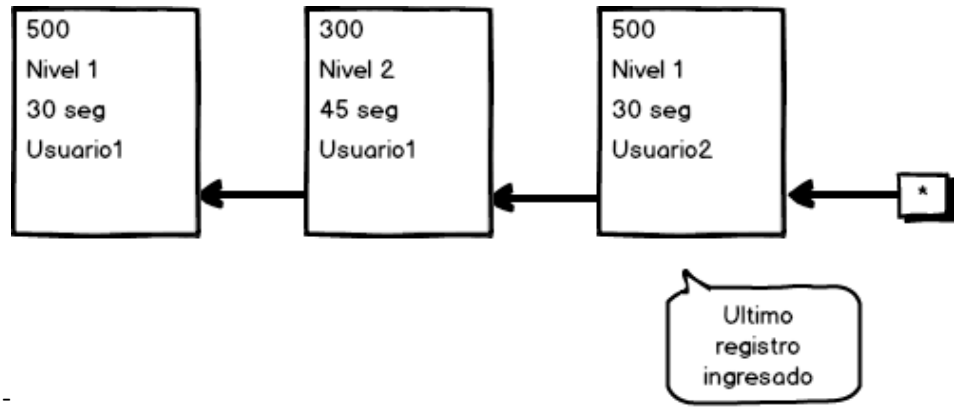

Figura 14. Lista de puntajes impares

- Implementación para carnet Par:
    - Matriz ortogonal – punteros hacia derecha y hacia arriba
    - Lista puntajes: Nodo cabeza será el primer registro ingresado, punteros siguientes

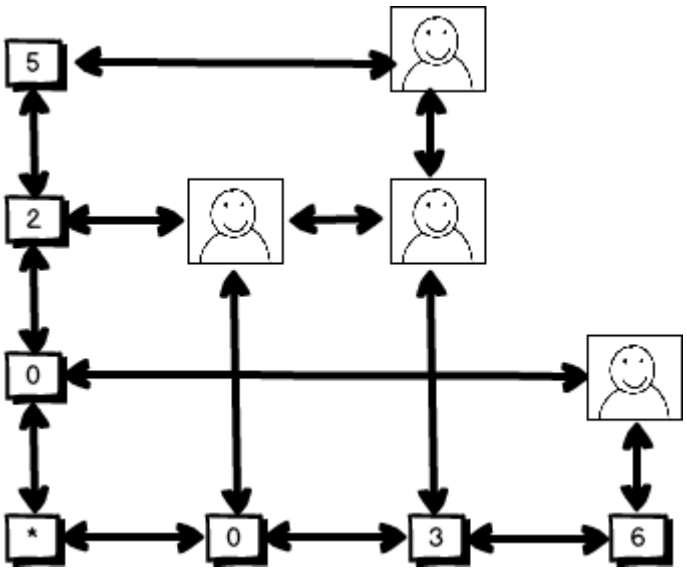

Figura 15. Matriz ortogonal pares

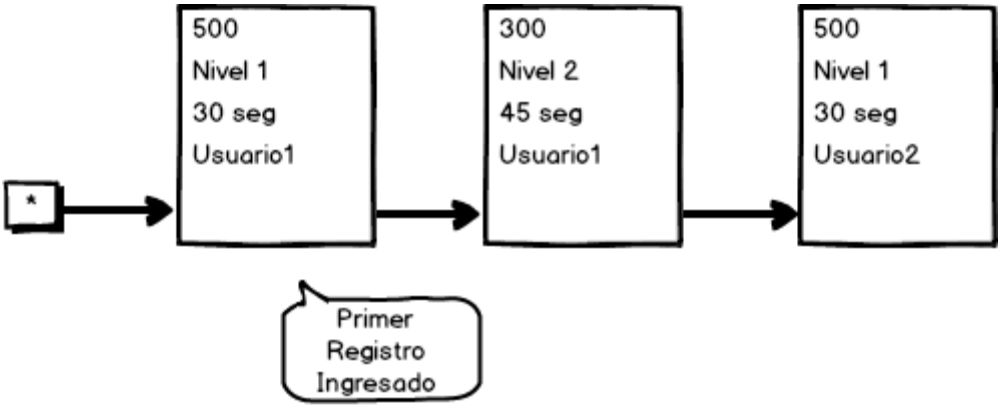

Figura 16. Lista puntajes pares

- Se verificara que se hayan utilizados los punteros para las estructuras indicadas, en el inciso anterior

ENTREGABLES

- Código Fuente
- Ejecutable
- Manual Técnico
- Manual Usuario

FECHA Y MODO DE ENTREGA

- Viernes 16 de Marzo de 2018 antes de las 23:59, tendrán una automáticamente una nota igual a 0
- Se deben de enviar los entregables vía Dropbox, en un comprimido con el nombre: [EDD]Proyecto1_#carnet (e.g. [EDD]Proyecto1_201318613)
- Dropbox: https://www.dropbox.com/request/z6pQA4CgH0bcL94rW95y

Dudas y Consultas

- Se responderán dudas en horario de DSI o vía correo mediante el asunto
  "[EDD]Duda Proyecto 1"

Requerimientos Mínimos

- Servidor: Creación de Niveles, animación de enemigos, pila de enemigos, consola, fin de juego
- Cliente Completo
- Interacción del juego entre cliente y servidor

Penalizaciones

- Si ocurre un NULL Pointer Exception, la nota obtenida será 0 automáticamente
- Si se cierra la aplicación repentinamente, la nota obtenida será 0 automáticamente
- Las representaciones graficas de las estructuras son esenciales para verificar si se realizó de manera correcta la estructura, por lo que si no se representan de esta manera NO se calificara
- Si NO se realizan las estructuras de la manera en que se indica en la sección Observaciones la nota obtenida será 0 automáticamente.
- Si se detecta el uso de librerías como QStack, QQueue, QList, etc, para C/C++ o el uso de LinkedList, Stack, Queue, o similares para Java, la nota obtenida será 0 automáticamente
- Las representaciones graficas deberán de mostrarse integradas en las GUI, y realizadas utilizando Graphviz, de no ser así NO se calificara

***Copias totales o parciales, tendrán nota de 0 puntos, serán reportadas al catedrático y a la Escuela de Ciencias y Sistemas FI-USAC**