**LABERITNOS Y FANTASMAS**

**SECCIONES/MODULOS**

* Un menú, con Jugar, Ver ranking y Salir, esta sale fácil, puede ir en el main y llamarse a cada función desde ahí.
* Config.txt, datos:
  + # paredes
  + . caminos
  + E entrada (inicio, siempre en un borde)
  + S salida (en el borde opuesto)
  + F fantasmas
  + P premios
  + V vidas extra

Se va a necesitar una función que lea esto, el txt se puede crear a mano supongo, lo que importa es la función que lee el txt de config al oprimir jugar (leer\_config()). También se necesitaría una estructura que guarde todos esos datos juntos para hacerla más fácil.

* Un tablero con las configuraciones cargadas (char tablero \*\*, siempre se vuelve a tópicos). Una función que cree el tablero sería lo mejor, se puede arrancar creando laberintos aleatorios que tengan entrada + salida + recorridos posibles, después ir agregando fantasmas en posiciones válidas (al menos dos casillas alejados de la entrada y salida diría). Tendría que haber un tamaño mínimo de tablero, con un 4x4 no hacemos nada, algo que valide tam mínimo.
* Lógica de movimiento:
  + Una para el jugador, movimientos WASD se encolan, ya habrá que ver cómo hacemos, diría de preguntar a qué se refiere porque muy bien no entiendo qué es que los movimientos se encolan.
  + Lógica de fantasmas, se mueven a una posición contigua válida al azar o cercana al jugador. Primero se podría hacer movimiento random para ir probando. -> INVESTIGAR ALGUNA IA PSEUDO PACMAN
  + Los premios no se mueven, pero hay que decidir cómo spawnean.
* Registro de movimientos: Ir guardando movimientos en una cola o vector y después los mostramos al final, no sé si tendrán que persistir los movimientos de cada partida, habría que guardarlos en distintos txt en ese caso.
* Sockets:
  + Recordarme que el tp de SO lo hice con sockets.
  + Se usa un binario para guardar el ranking.
  + NO listas, aclara que se hace con árboles el ranking.
  + Hay que hacer un índice, osea ordenar por campo puntuación.
  + Se guarda en el ranking: Usuario, puntuación, cantidad de movimientos en cada partida.
    - ¿Hay que acumular los datos de cada usuario entre partidas o admitir duplicados y que se guarde un registro por partida? Recordar preguntar eso al profe.
  + Encolar clientes (mandemos máx 1 cliente simultáneo para que no haya quilombos de sincronización, si se intenta conectar un segundo cliente, el programa lo vuela).
  + ¿Un posible modo offline que deje jugar sin mostrar ni actualizar ranking si no se puede conectar? A debatir.
* El maldito SDL, alguien va a tener que aprender a usarlo, no es sugerencia.

Espacio para sugerencias de mejoras, esto se debate entre todos y si cuesta mucho implementar se vuela:

* Dificultades.

**FUNCIONES**

Son las principales que se me van ocurriendo mientras hago este documento, pueden faltar o sobrar.

Todas en snake\_case, sigamos un único formato.

* salir()
* ver\_ranking()
* jugar()
* leer\_config()
* crear\_tablero()
* cargar\_laberinto()
* renderizar\_texto()
* renderizar\_laberinto()
* ingresar\_nombre()
* fantasmas()
* jugador\_juega()
* conectar\_servidor()
* leer\_archivo\_ranking() (creo que una primitiva de árbol ya lo hace)
* mostrar\_ranking()
* cerrar\_servidor()
* jugador\_gana()
* jugador\_pierde()
* fin\_partida()
* Primitivas de cola
* Primitivas de árbol, incluyendo las de índices y archivos binarios
* Primitivas de lista
* etc.

**ESTRUCTURAS**

* Jugador(nombre, puntaje, pos\_x, pos\_y)
* Fantasma(pos\_x, pos\_y)
* Configuracion(filas, columnas, vidas\_iniciales, max\_fantasmas, max\_premios, max\_vidas\_extra)
* Tablero(fila, columna)
* Indice
* UserRanking(nombre, cant\_movimientos, puntos)