Universidad autónoma de Aguascalientes.

Centro de Ciencias Básicas

Ingeniería en Sistemas Computacionales

Proyecto Final

"Memoria Extreem"

Aguascalientes, Ags.

Listas de actividades:

- Lógica de juego
- Lógica de simulación
- Lógica del registro
- Lógica de mantenimiento
- Implementación de letras grandes
- Flechas
- Creación de funciones
- Centrado de textos
- Lógica de pantallas
- Pruebas extensivas en Linux
- Lógica de Menús

Conclusión:

Disfrute mucho crea la lógica del juego, lo que más me gusto hacer fue la simulación y el juego, reportes y mantenimiento fue algo sencillo, pero realmente me encanto todo lo que logre trabajar haciendo el proyecto, ciertamente hay muchas cosas todavía que podrían cambiarse y optimizarse, yo utilice una matriz booleana que guardara si la combinación fue fallida o no, y al buscar la casilla la siga buscando si es que la combinación fue fallida.

Algo que también disfrute fue hacer los menús, algo que le agrega rapidez y fluidez al juego, ciertamente se ve asombroso como con la flechita puedes seleccionar que es lo que tú quieres, y que siempre haya una opción salir si es que ya no deseas seguir.

También algo a destacar fue la utilización de git en el proyecto, esto nos ayudó muchísimo ya que antes en el primer proyecto del primer semestre, solíamos pasarnos el código por whattsapp, y al final con tantos nombres, no sabíamos ni cual era el actual, pero todo cambio cuando usamos git, fue algo que nos facilitó muchísimas cosas y nos fue de mucha ayuda.

En cuanto al curso, hubo muchas tareas y cosas que me hubiera gustado entregar, el curso fue divertido y de cierta manera siempre había algo más que aprender y que practicar, eso es lo que me gusta de esta materia, al igual que muchas otras, no logras nada si no te pones a hacerlo tú mismo y comienzas a practicar y a hacer pruebas de código o de escritorio.

Listas de actividades:

- Investigación de librerías
- Logo de la uaa
- Uso de enums
- Organización de código
- Debug del programa
- Códigos de errores
- Lógica de validaciones
- recuadro de colores
- creación de funciones
- creación de librerías
- makefile
- Funciones de sonido por sistema
- Proyecto en devcpp y codeblocks
- Lógica de centrado
- Pruebas extensivas en Windows
- Búsquedas de referencias

Conclusión:

Puedo concluir que personalmente me puse como meta buscar la forma en la que nuestro proyecto funcionara no solo en Windows y Linux, sino que en cualquier otro sistema operativo similar a unix, como lo seria MacOs y cualquier BSD como OpenBSD y Freebsd, también mientras hacíamos el proyecto aprendí de orden de código, logica secuencial, gestión de versiones y ramas en git, teclear más rápido, liderazgo, gestión de proyecto, ver errores antes que sucedieran y uso de sistemas operativos, entre otras muchas cosas.

Una de las cosas que aprendí fuera de la programación, fue aprender a trabajar en equipo y a tener control de mi tiempo y ver que necesitaba el proyecto para hacer un espacio de tiempo para poder avanzar al proyecto.

También para el proyecto me asegure que la mayoría de los temas vistos en el curso de programación hayan sido usados, solo uno que otro tema no está implementado como la recursión o la unión, pero de echo al final de ver el tema de enums, decidí implementarlo, también la parte de memoria dinámica y estática fue uno de los que considere al revisar el programa, ya que al final, el programa usa en promedio menos de 5mb de ram, en conclusión nos quedó muy bien el proyecto y me siento orgulloso de lo que hicimos.

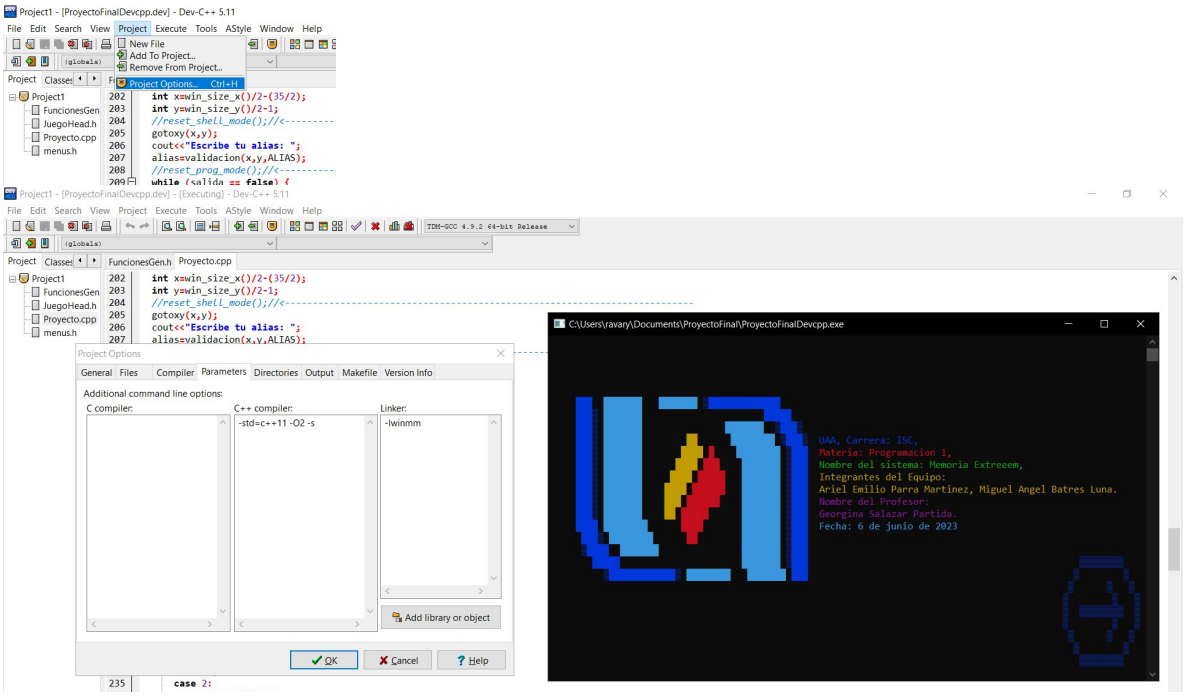
Conclusión General:

En general, es algo difícil aclarar quien hizo cada cosa, entre los dos nos complementamos, cuando alguien de los dos crea un código, luego el otro lo utiliza para algo más, cabe destacar que si uno de los dos, no hubiera recibido ayuda del otro, no hubiéramos logrado lo que hemos logrado, algunas veces había problemas con ciertos códigos o con la lógica de los mismos y en especial con las decisiones creativas y gráficas, pero aun así siempre llegábamos a un acuerdo y se podría decir que ambos trabajamos lo mismo y dimos todo nuestro esfuerzo y dedicación.

Datos Extras del Proyecto:

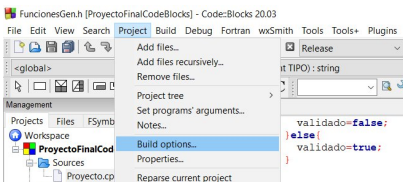
Para este Proyecto creamos un Proyecto de Devc++ llamado ProyectoFinalDevcpp.dev

Para esto ocupa unas opciones de proyectos

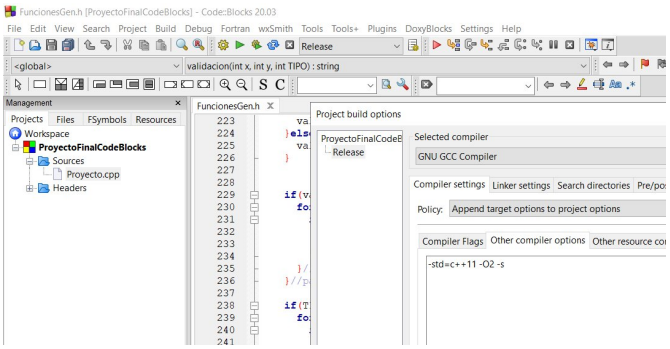


En especial ciertos para metros del compilador (g++), la más importante es: -std=c++11 ya que esta nos deja usar funciones de C++11 como las funciones de std::string, y también el parámetro de link: -lwinmm , ya que este nos deja utilizar la función playsound en windows, los parámetros -O2 -s, son para optimizar y reducir en tamaño el binario resultante respectivamente.

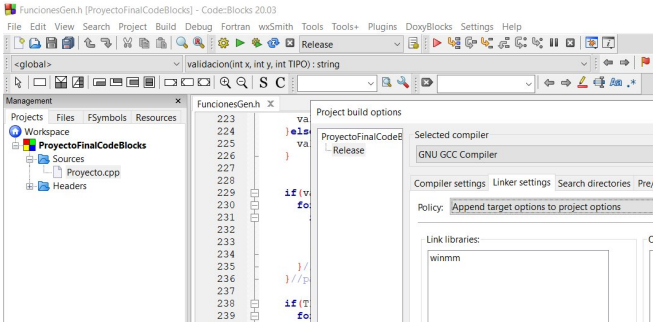
También creamos un proyecto de CodeBlocks llamado ProyectoFinalCodeBlocks.cbp, este ocupa los mismos parámetros que el anterior, solo que aquí se ponen en “build options”

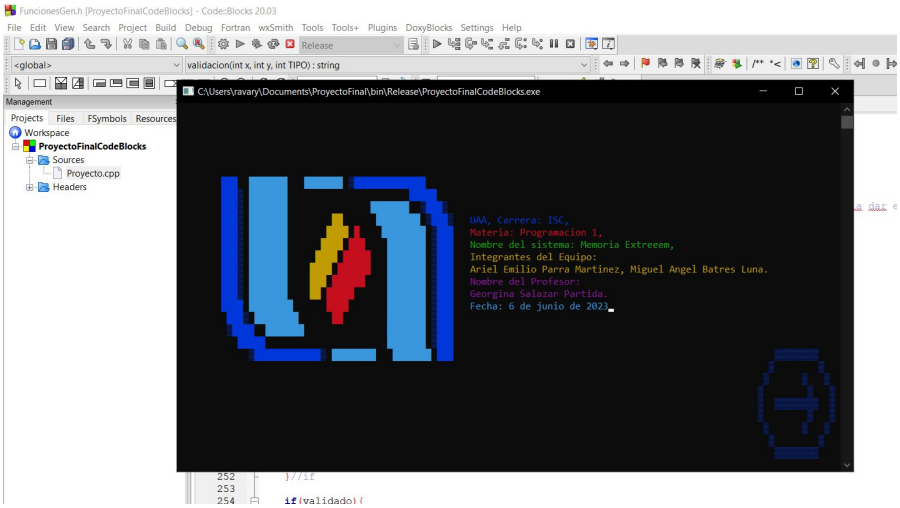


Y se ponen los parámetros personalizados en “other compiler options”



Y el linker -lwinmm, se pone solo como winmm



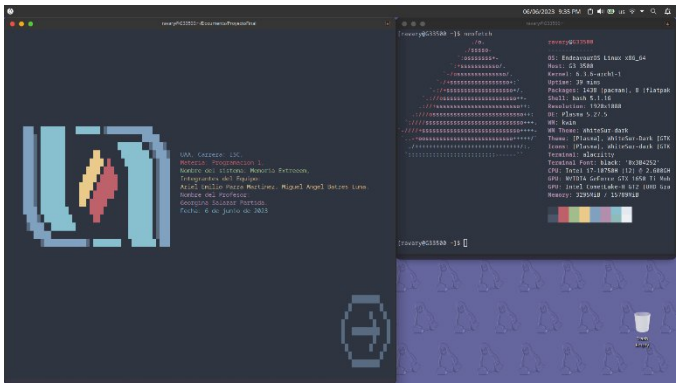


Cabe mencionar que para la grabación del video Miguel estaba enfermo, por lo que tuvimos que grabar todo en una sola tarde para que el pudiera descansar, pero tuvimos muchos problemas técnicos para empezar a grabar, ya que microsft teams, no podía grabar de manera correcta en Linux, por lo que recurrimos a grabar dentro de en Windows usando el subsistema de Linux para Windows (WSL), ya que el proyecto se puede llegar a ver algo trabado en Windows, y en Linux se ve mejor, aunque la implantación de WSL obliga a usar iso C++17, pero esto no afecta el programa ya que en Windows aun así ocupaba el iso C++11, aunque hasta después de grabar vimos los pequeños problemas técnicos con el audio, estos pasan debido a que WSL utiliza el sonido de Windows, pero estos problemas técnicos no se encuentran en los demás sistemas operativos.

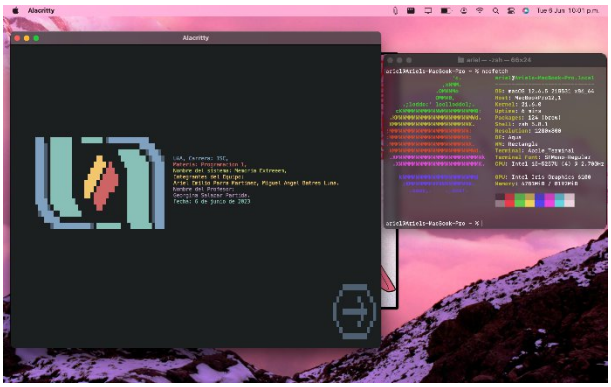
Para este proyecto me esforcé en hacer compatibles la mayoría de los sistemas operativos, aunque solo probe seis de los más usados y populares, los cuales listare y comentare sus “problemas”, también anexo imágenes de las pruebas.

1. Linux (nativo): ningún problema
2. MacOs (nativo): ningún problema
3. OpenBSD (emulado): ningún problema
4. FreeBSD (emulado): ocupa correrse en el shell bash, ocupa instalar la librería ncurses y el programa sox para el audio, usando el comando play
5. Windows 10 (nativo): el juego da tirones constantes en cmd, y la pantalla se limpia de manera inconcisa en cualquier terminal incluso usando powershell
6. Linux (WSL): el sonido puede dar errores, ocupa instalar la libreria ncurses y el programa ffmpeg para el audio, usando el comando ffplay

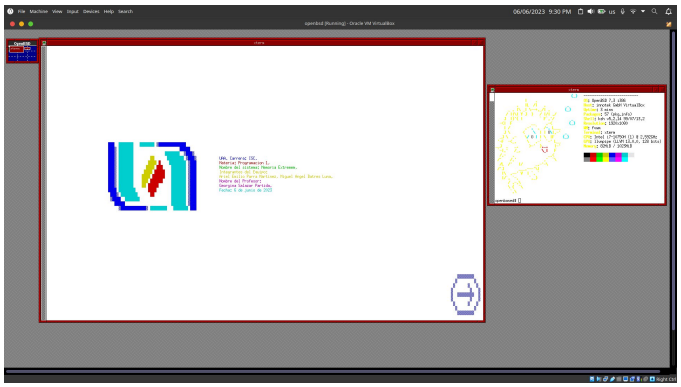
Linux



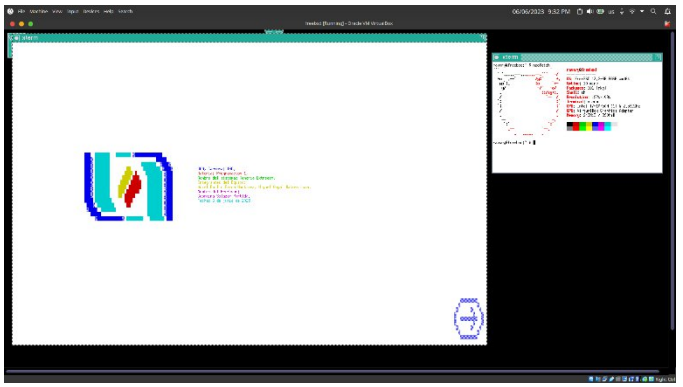
MacOs



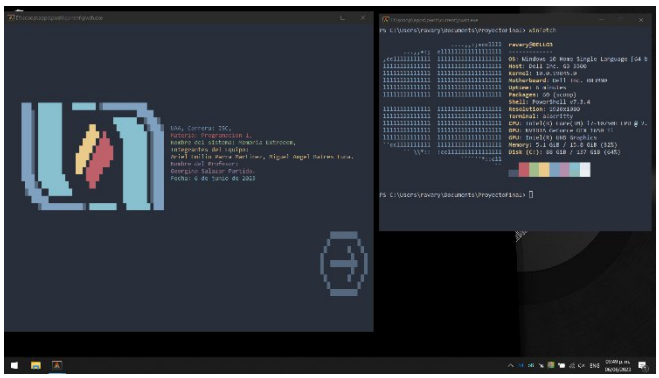
OpenBSD



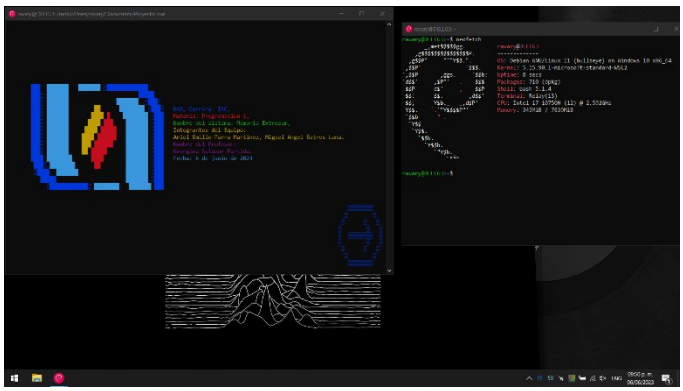
FreeBSD



Windows 10



Linux (WSL)



Referencias:

/*sonidos*/

pop.wav recuperado de: https://soundbible.com/2067-Blop.html

sonidos de error.wav y exito.wav desarrollados con: https://sfxr.me/

song.wav recuperado de BossMain.wav en la pagina:
https://opengameart.org/content/nes-shooter-music-5-tracks-3-jingles

win.wav recuperado de Victory.wav En la pagina :
https://opengameart.org/content/victory

/*Graficos*/

- convertir png uaa a unicode/ansi: https://manytools.org/hacker-tools/convert-image-to-ansi-art/
- logo de la uaa: https://upload.wikimedia.org/wikipedia/commons/thumb/f/fe/Logo_de_la_Universidad_Aut%C3%B3noma_de_Aguascalientes.svg/2415px-Logo_de_la_Universidad_Aut%C3%B3noma_de_Aguascalientes.svg.png
- palabras con unicodes/ansi: https://fsymbols.com/

/*Referencias para codigos*/

- convertir string a char *, recuperado de: https://stackoverflow.com/questions/347949/how-to-convert-a-stdstring-to-const-char-or-char
- creación headers: https://learn.microsoft.com/en-us/cpp/cpp/header-files-cpp?view=msvc-170
- números ascii para validación entrada de strings https://en.cppreference.com/w/cpp/language/ascii
- uso de libreria ctime: https://stackoverflow.com/questions/997946/how-to-get-current-time-and-date-in-c y https://cplusplus.com/reference/ctime/tm/
- uso de using std::, en vez de using namespace: https://stackoverflow.com/questions/1452721/why-is-using-namespace-std-considered-bad-practice

/*códigos portables/compatibles*/

- detectar linux dentro de wsl: https://en.cppreference.com/w/cpp/utility/program/getenv https://github.com/microsoft/WSL/issues/8406
- detectar sistemas operativos: https://github.com/cpredef/predef
- imprimir caracteres unicode en windows, recuperado de: https://www.appsloveworld.com/cplus/100/31/properly-print-utf8-characters-in-windows-console

- make file windows/nix por Paul Hutchinson: https://stackoverflow.com/questions/4058840/makefile-that-distinguishes-between-windows-and-unix-like-systems
- obtener tamaño de la consola en ncurses, recuperado de: https://stackoverflow.com/questions/1811955/ncurses-terminal-size
- obtener tamaño de la consola en windows, recuperado de: https://stackoverflow.com/questions/6812224/getting-terminal-size-in-c-for-windows/12642749#12642749
- plantilla de gotoxy para linux recuperado de: https://stackoverflow.com/questions/7581347/gotoxy-function-with-c-linux-unix
- Plantilla de gotoxy para windows recuperado de: https://stackoverflow.com/questions/55635791/how-to-use-function-gotoxyint-x-int-y
- reproducir sonido en openbsd recuperado de: https://man.openbsd.org/aucat.1
- reproducir sonido macos recuperado de: https://superuser.com/questions/598783/play-sound-on-mac-terminal
- uso de librería ncurses: https://invisible-island.net/ncurses/ncurses-intro.html
- uso de playsound, recuperado de: https://learn.microsoft.com/en-us/previous-versions/dd743680(v=vs.85)