


The Game Of Life

A játék építése Windows és Linux-os rendszerekre:

- Követelmények
 - [cmake](#)
 - Ha Linuxon [make](#)
 - Ha Windowson akkor [Visual Studio](#)
- A program építése és futtatása
 - Linux
 1. `cmake .`
 2. `make`
 3. `./TheGameOfLife`
 - Windows
 1. `cmake .`
 2. Nyisd meg a `.sln` projectet.
 3. Solution Explorer -> Configure Startup Projects -> Single startup project: The Game Of Life
 4. Nyomd meg a  gombot.

A játék menete

- Amikor megjelent a terminál ablak akkor mehet a játék.
- A program egy Title kiírás után a program megkérdezi, hogy játékot létrehozni vagy betölteni szeretne, vagy kilépni a programból.
 - Új játékmenet kiválasztásakor
 - A program megkérdezi a mentés nevét és a játék méretét is.
 - Utána a játék kinyomtatja a kezdetleges játékmezőt
 - Játék betöltésekor
 - A program kifogja írni a játékosnak az összes mentését.
 - A felhasználó ki fogja tudni választani a mentést és ez megmutatja a játékmezőn is.
- A Játék
 - A következő lépés gombbal. A következő lépést kifogja számolni is kifogja írni a kiejtőre.
 - A játékos kifogja tudni választani egy cursor segítségével, hogy melyik cellát szeretné megölni vagy életre kelteni.
 - Random gombbal a felhasználó randomizálni fogja tudni a játék összes celláját.
 - A törlés gombbal a felhasználó törölni fogja tudni az egész játékmezőt.
 - Mentés gombbal a felhasználó menteni fogja tudni az adott mentését.
 - Vissza gombbal: ha még a felhasználó nem mentette el a játékot ezt megfogja kérdezni, és visszamegy a fő menübe.
 - Kilépésnél: ha még a felhasználó nem mentette el a játékot ezt megfogja kérdezni, és kilép az egész programból.

A játék háttere.

src/game/CharLogicHandler.c

- `char * CpyStr(char * str)`
 - **Malloc -ot használ.**
 - Lefogja másolni az adott Stringet egy új memória területre.

src/game/FileHandler.c

- `char * MakePath(char * str, bool catFileFormat)`
 - **Malloc -ot használ.**
 - Mikor filenevet kap egy új Stringet fog létrehozni és tartalmazni fogja a „Relative Path” ot a file elérését.
 - Ha „catFileFormat” true akkor .csv file format -ot hozzáfűzi.

- `int InitSaveFolder()`
 - `int`: Az egy hibajelző kimenet lesz.
 - Létre fogja hozni a mappát, amibe a mentéseket rakjuk.
- `bool FileNameHasBadChar(char str[])`
 - Egy file nevet fog kapni és megfogja alapítani, hogy invalid e. (nem tartalmazhat: <:>\"/|?*)
- `bool DoesFileExist(char * str)`
 - Filenevet kap és megfogja állapítani, hogy a file létezik e.
- `int SaveMatrixToFile(Matrix * matrix, char * str)`
 - `int`: Az egy hibajelző kimenet lesz.
 - Átfogja kapni a Mátrix adatát és a file nevét és ezt elfogja menteni a str nevű fileba.
- `int GetSizeFromFile (SizeMatrix * size, GameSaveFiles * files, int select)`
 - `int`: Az egy hibajelző kimenet lesz.
 - Vissza fogja adni a mátrix méretét kiválasztott filéből.
 - Megfogja kapni a file neveket és melyiket szeretnénk megnyitni. És visszaadja a játék méretét size pointeren keresztül.
 - <- használni kell (értéket ad vissza) (SizeMatrix * size)
- `int LoadGameFromFile (Game * game, GameSaveFiles * files, int select)`
 - Befogja töltetni a játékot fileból.
 - Megfogja kapni a játék instance -ét és a filenevek közül melyiket szeretnénk betölteni.
- `int GetSaveFiles (GameSaveFiles ** files)`
 - `int`: Az egy hibajelző kimenet lesz.
 - **Malloc -ot használ.**
 - Megfogja nézni a mappába milyen mentések vannak és a GameSaveFiles struktúrába elmenti és ennek visszaadja a * ját.
 - <- használni kell (értéket ad vissza)
- `void DestroyStructSaveFiles (GameSaveFiles * files)`
 - Törölni fogja a GameSaveFiles létrehozott adat struktúráját.

src/game/FileHandlerStruct.h (Header file)

- `#define`
 - Tartalmazza a mentés mappa nevét és file formátumát és file név hosszúságát
- `struct GameFileProperties`
 - Lefogja írni a játék közben használt filet és hogy mentette-e a felhasználó.
- `struct GameSaveFiles`
 - Lefogja írni a mentés mappában szereplő file nevét és számát.

src/game/GameLogic.c

- `Matrix * InitializeMatrix (SizeMatrix size)`
 - **Malloc -ot használ.**
 - Ha malloc failed akkor a return az NULL.
 - Létrehoz egy megadott méretű Mátrixot.
- `void DestroyMatrix (Matrix * matrix)`
 - Kitörli a `InitializeMatrix` által létrehozott memóriát.
- `Game * InitializeGame ()`
 - **Malloc -ot használ.**
 - Létrehozza a Game -et és default adatértékeit.
- `void DestroyGame (Game * game)`
 - Kitörli a `InitializeGame` által létrehozott memóriát.

- `void DeleteGameData (Game * game)`
 - Kifogja törölni a game struct -bol az adatot és default -ra állítja.
 - Nem fogja free -elni a memória területet.
- `int FindNeighbors (Matrix * matrix, Point point)`
 - Ha kap egy pont -ot akkor mátrix -on belül a körülötte levő élő cellákat összefogja számítani és visszaadni.
- `int NextStep (Matrix ** matrix)`
 - Kiszámítja az új lépést a játéknak és felül írja a Mátrix -nak a pointerjét.
 - A régi mátrixot kitörli
 - & <- használni kell (értéket ad vissza)

src/game/GameLogic.h (Header file)

- `struct SizeMatrix`
 - Lefogja írni a 2 dimenziós mátrix x és y méretét.
- `struct GameSaveFiles`
 - Lefogja írni az összes file-t, ami Save mappában található, hogy hány db van és a file neveiket.

src/game/IntLogicHandler.c

- `int IntDigitSize (int num)`
 - Kifogja számolni, hogy egy számba hány db szám áll.
- `int IntFindTheLargest (int const nums[], int numCount)`
 - Egy nums array be megtalálja a keresett elemet és visszaadja a keresett értékét.
- `int IntGetDigitWithIndex (int num, int index)`
 - Egy számot indexelni fogunk tudni. Pl num: 123 index:0 return 1 ha index az 1 akkor return 2

src/game/Random.c

- `void InitRandom ()`
 - Befogja állítani a srand -ot „good enough” seed el.

src/terminal/GameSession.c

- `int LoadMenu (MenuOption menuOption, Game * game)`
 - Enum által választható menü és game instance átadásával betölt egy menü pontot.
 - `enum { mainMenu, newGame, loadGame, mainGame, abortGame }`
- `void HandleMainMenu (Game * game)`
 - Lekezeli a Main Menü betöltését és Action -jait
- `void HandleNewGame (Game * game)`
 - Lekezeli az új játék Menü betöltését és Action -jait
- `void HandleLoadGame (Game * game)`
 - Lekezeli a játék betöltés Menüt és Action -jait
- `void HandleMainGame (Game * game)`
 - Lekezeli a játék Menü betöltését és Action -jait
- `void HandleGameNextStep (Game * game)`
 - Lekezeli a játék következő lépését
- `void HandleGameModify (Game * game)`
 - Lekezeli a játék módosítás Menü betöltését és Action -jait
- `void HandleGameRandomize (Game * game)`
 - Lekezeli a játék randomizáció Menü betöltését és Action -jait
- `void HandleGameClear (Game * game)`
 - Lekezeli a játék kijelző törlését Menü betöltését és Action -jait

- `void HandleGameSave (Game * game)`
 - Lekezeli a játék mentését
- `void HandleDoYouWantToSave (Game * game)`
 - Lekezeli, hogy a felhasználó menteni akarja-e a mentését.
- `void HandleGameBack (Game * game)`
 - Lekezeli a játék Main Menübe visszatérését.
- `void HandleGameQuit (Game * game)`
 - Lekezeli a játék kilépését.

src/terminal/GameSession.h (Header file)

- `#define`
 - Leírja a terminál Max x és y méretét.
 - És Windows -on és Linux on a Arrow Key -eket.
- `enum { mainMenu, newGame, loadGame, mainGame, abortGame } MenuOption`
 - Lekezeli a menü opciókat.

src/terminal/PrintHandler.c

- `void PrintBoxTop (int width, int indent)`
 - A játék mátrix -nek a tetejét kifogja nyomtatni. Pl „+-----+”
- `void PrintNumbersVertically (int length, int indent)`
 - Kapni fog egy méretet és kifogja nyomtatni az összes számot addig sorrendbe függőlegesen.
- `void PrintMatrixBoard (Matrix * matrix)`
 - Ki nyomtata a mátrix -ot a kijelzőre stílussal.
- `void PrintMatrixBoardWithPoint (Matrix * matrix, Point point)`
 - Ugyan az fogja csinálni, mint `PrintMatrixBoard` csak egy cursort még lefog rakni a megadott pontra.
- `void ClearScr ()`
 - Kitöröl mindent a ki jelzőről.
- `void PrintHeader (char * str)`
 - A menü nek a fejlécét fogja kinyomtatni.
- `void PrintLogo (WinSize winSize)`
 - A logót kifogja nyitadni, ha van elég hely a kijelzőn.
 - A kijező méretet fogja megkapni.
- `void PrintMainMenu ()`
 - Kifogja nyomtatni a main menüt.
- `void PrintFiles (GameSaveFiles * files)`
 - Kifogja nyomtatni az összes Filet, ami a Save Mappában található nevét a kinézőre stílus al.
- `void PrintBack ()`
 - Kifogja nyomtatni a Back gomb nak a szövegét.
- `void PrintGameMenu ()`
 - A game menüt fogja kinyomtatni.
- `void PrintGameWasSaved ()`
 - Kijelzés a mentés sikerességére.

src/terminal/PromptHandler.c

- `void PurgeStdin ()`
 - Ha scanf túlcscordul Pl (ha számot kér és betűt kap) akkor a \n ig kiveszünk mindent a Stdin ból.
- `char ReadChar ()`
 - 1 db char -t beolvas enter nélkül.

- `int PromptMainMenu()`
 - Beolvassa a main menü választ a felhasználótól.
- `int PromptFileName(char * str)`
 - int: Az egy hibajelző kimenet lesz.
 - filenevet fogja beolvasni és visszaadja str -en keresztül.
- `SizeMatrix PromptMatrixSize(Game * game)`
 - A mátrix méretét fogja megkérdezni a felhasználótól.
- `int PromptFileLoad(int * select, GameSaveFiles * files, Game * game)`
 - int: Az egy hibajelző kimenet lesz.
 - select fogja visszaadni a kiválasztott mentést.
- `int PromptBack()`
 - A vissza gombot fogja megkérdezni a felhasználótól.
- `int PromptGameMenu()`
 - A játék menüből fog tudni választani a játékos.
- `int PromptYesNo(bool * YesNo, char ask[])`
 - int: Az egy hibajelző kimenet lesz.
 - A function kap egy ki írandó szöveget és a felhasználó eltudja dönteni, hogy igent vagy nem választ.
 - YesNo ba fogja visszaadni a választ.
- `int PromptCursor(Point * cursor, SizeMatrix size)`
 - int: Az egy státusz kimenet lesz.
 - Arrow key -eket fogja nézni. És cursor -on keresztül fogja visszaadni.

src/terminal/StyleHandler.c

- `void MoveCursorUp(int steps)`
 - A kurzor felfele nyomása x alkalom al.
- `void EraseInLine()`
 - Sor kitörlése.
- `void AnsiResetAll()`
 - Minden Stílus visszaállítása.
- `void AnsiColorBlack()`
 - Fekete Szín beállítása.
- `void AnsiColorRed()`
 - Piros Szín beállítása.
- `void AnsiColorGreen()`
 - Zöld Szín beállítása.
- `void AnsiColorYellow()`
 - Sárga Szín beállítása.
- `AnsiColorBlue()`
 - Kék Szín beállítása.
- `void AnsiColorMagenta()`
 - Magenta Szín beállítása.
- `void AnsiColorCyan()`
 - Cián Szín beállítása.
- `void AnsiColorWhite()`
 - Fehér Szín beállítása.
- `void AnsiBackgroundBlack()`
 - Fekete Háttér Szín beállítása.
- `void AnsiBackgroundRed()`
 - Piros Háttér Szín beállítása.
- `void AnsiBackgroundGreen()`
 - Zöld Háttér Szín beállítása.
- `void AnsiBackgroundYellow()`
 - Sárga Háttér Szín beállítása.

- `void AnsiBackgroundBlue()`
 - Kék Háttér Szín beállítása.
- `void AnsiBackgroundMagenta()`
 - Magenta Háttér Szín beállítása.
- `void AnsiBackgroundCyan()`
 - Cián Háttér Szín beállítása.
- `void AnsiBackgroundWhite()`
 - Fehér Háttér Szín beállítása.
- `void StyleBold()`
 - Kövér stílus beállítása.
- `void StyleItalic()`
 - Dőlt stílus beállítása.
- `void StyleUnderline()`
 - Aláhúzás stílus beállítása.

src/terminal/WindowSize.c

- `WinSize GetWindowSize()`
 - A terminál ablak méret lekérdezése.
- `bool IsXTooBig(WinSize winSize, size_t x)`
 - Megkapja a terminál méretét és megnézi hogy x túl nagy e.
- `bool IsYTooBig(WinSize winSize, size_t y)`
 - Megkapja a terminál méretét és megnézi, hogy y túl nagy e.

src/terminal/WindowSize.h (Header file)


- `struct WinSize`
 - lefoglalja a terminál méretét

src/util/Stringify.h (Header file)

ha kap egy `#define` ot akkor át fogja alakítani string é

src/util/Utils.c

- `void AbortMsg(char str[])`
 - Ki ír egy hibaüzenetet a kijelzőre.
- `void SleepTime(int time)`
 - Megadott ideig aludtatni fogja a programot.

 Nyitott forráskód, amit a programba használtam.

[Github Workflow](#)

[Játék Logo](#)

.gitignore [cmake](#) [visual studio](#)