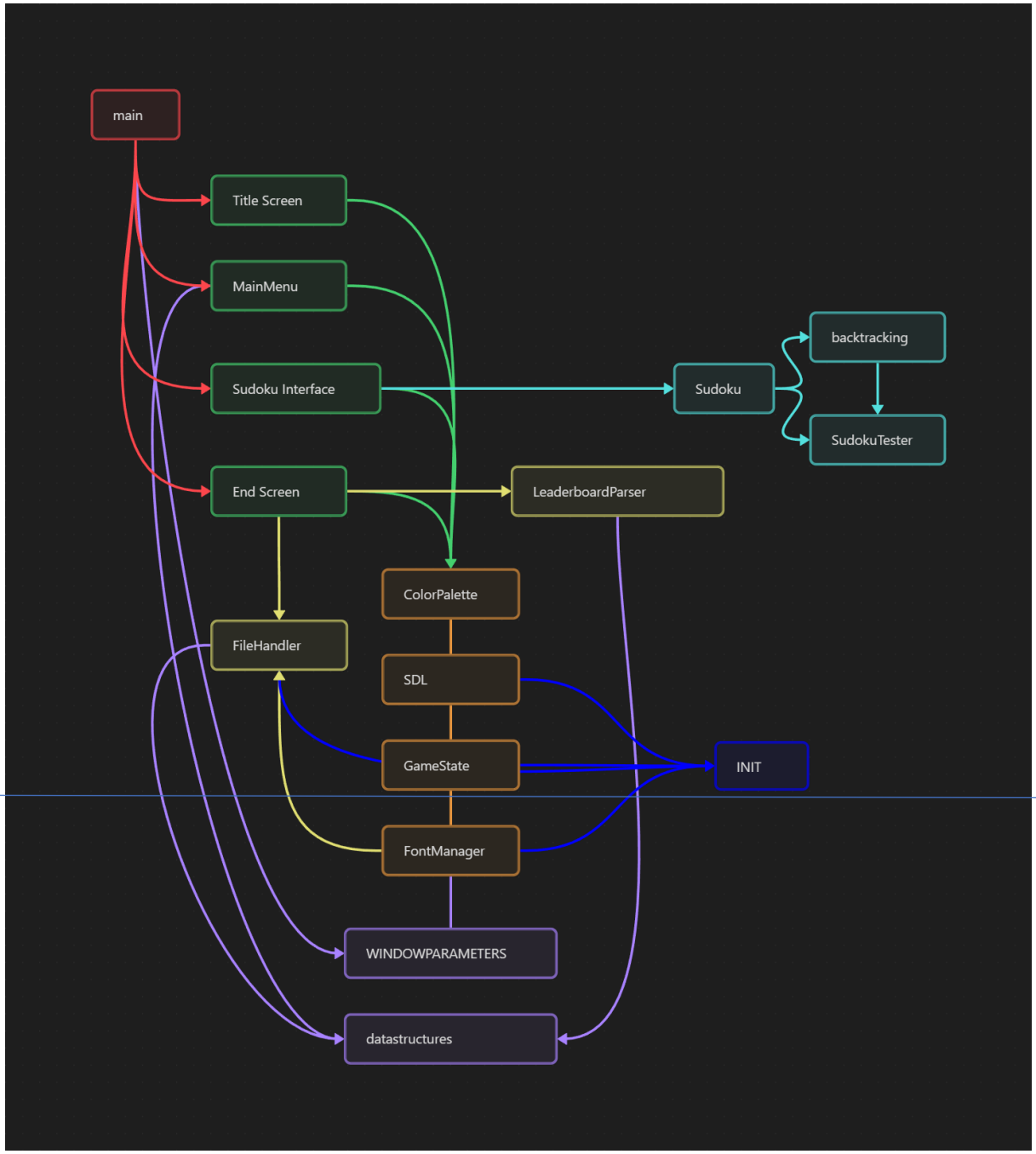
HQ21TL
Urbán Kristóf Tamás Dokumentáció V1.2

SUDOKU

Alap adatok:

- A program a mingw64 gcc 13.2.0 compiler-hez készült.
- Használja az:
  - SDL2 (v 2.28.4) grafikus könyvtárat.
  - SDL2_image (v 2.6.3) SDL2 kiegészítőt.
  - SDL2_ttf (v 2.20.2) SDL2 kiegészítőt.
- Az applikáció futtatásához szükséges:
  - SDL2.dll
  - SDL2_image.dll
  - SDL2_ttf.dll
- Az applikáció felhasználja: (Ahol nincs a dokumentációban forrás, a data/assets/sources.txt fájlban megtalálható.)
  - data/assets/font.ttf - Ez a BebasNeue_Regular font.
  - data/assets/icon.png
- Az applikáció futtatáshoz felhasználja, de nem feltétel:
  - data/savedata.bin
  - data/leaderboard.bin
- Aktuális forrás fájlok:
  - backtracking c/h
  - ColorPalette c/h
  - datastructures h
  - debugmalloc h
  - end_screen c/h
  - ErrorHandler c/h
  - file_handler c/h
  - FontManager c/h
  - GameState c/h
  - Init c/h
  - Leaderboard_parser c/h
  - main c/h
  - main_menu c/h
  - Sudoku c/h
  - Sudoku_tester c/h
  - SudokuInterface c/h
  - Title_screen c/h
  - WINDOW_PARAMETERS h
- Include és lib mappák tartalma.

A program hivatkozási ábrája.



ErrorHandler: Az ErrorHandler szerepe, hogy gyors hibakezelést nyújt. Jelenleg érzékeli a memória foglalás hibákat, és az SDL pointer és hibakód hibákat.

Init: Az Init az SDL könyvtár és kiegészítő könyvtásainak inicializálására ad gyors, hibakezelt lehetőséget.

file_handler:	Filebeolvasás és írás, alapértelmezett file létrehozás, asset betöltés.
GameState:	A játékállapotot tárolja.
FontManager:	A betűtípust kezeli.
ColorPalette:	A SDL színeket tárolja egyszerű hívásra.
main_menu:	A főmenü funkciót tartalmazza.
SudokuInterface:	Ez adja a sudoku játék és az Interfész között a kapcsolatot, és a játék felületének funkcióit.
Sudoku:	A játék fő programja, ezzel kommunikál a SudokuInterface.
Backtracking:	Ez az algoritmus generálja a megoldott táblát, majd ebből vesz ki a program cellákat a megoldatlan tábla létrehozásához.
Sudoku_tester:	Ez tartalmazza azokat az algoritmusokat, amikkel nézi a program a játék helyesen lett e megoldva.
End_screen:	Miután a játékos nyert, ezt a felület jelenik mega toplistával, új játék és visszalépési gombal.
Leaderboard_parser:	Felkészíti a nyers toplista adatsort megjelenítésre.
main:	Ez a bementi pont. Itt fut a fő ciklus, itt számolja a képkockák közt eltelt időt, itt értelmezi a felhasználótól kapott bemenetet és itt vált játékállapotok közt.

A program folyamata:

A program a main függvény hívásakor elkezdi a modulokat inicializálni, majd beolvassa a savedata.bin tartalmát, hogy visszaállítsa az előző folyamatban lévő állapotot. Amennyiben nem talál ilyen fájlt, vagy üres, akkor létrehoz egyet és feltöltő alapértelmezett értékekkel, majd ugyan ezekkel folytatja a futtatást.

A „mainloop” a program szíve, ez a ciklus csak az SDL_QUIT eseményre lép ki, ekkor menti a savedata.bin-be az aktuális adatait és kilép, mindent feltakarítva maga után.

A mainloop több fázisra van bontva első az eseménykezelő, ez a kilépés kérelmen kívül hallgat az ablak átméretezés, kattintás, billentyűnyomás eseményekre.

Ablak átméretezés eseten biztosítja, hogy nem léptük át alsó és felső határokat, valamint ez frissíti a globális ablak méret változókat.

Egy egér kattintás vagy billentyűnyomás esetén a program elmenti ezeket, majd a ciklus végén elveti.

A maradék fázisokat a játékállapot vezérlő határozza meg (GameSate). Ettől a ponttól kezdve ezek a játékállapotok határoznak meg mindent.

TitleScreen: Ebben az állapotban a titlescreen-t rendereli, majd a megadott idő után átvált a főmenüre. (Tudom, hogy egy nyelven kell írni, de ötletem sincs minek fordítsam.)

Főmenü: Ekkor a fűmenü feliratait, gombait beállításait rendereli, itt tudunk tábla mértet, nehézséget, valamint segítségeket állítani. A Play gomb megnyomásával átirányít a Sudoku interfészre, és meghívja a sudoku interfész generáló függvényét a beállított paraméterekkel.

Sudoku Interfész: Ez az állapot maga a játék. A főmenü által meghívott generáló által generált sudoku tábla itt megjelenítődik, valamint az eltelt idő, kilépés és segítség gombot is létrehozza. Hallgat kattintásokra és billentyű nyomásokra, ezeket nagyrészt továbbadja a sudokunak.

Sudoku: Ez nem játékállapot, hanem az interfésszel párhuzamosan létező programrész, ami minden sudokuval kapcsolatot dolgot vezérel. Fő funkciói a sudoku generálása, paraméterek visszaadása, cella írása, kiválasztása, hiba kezelés, valamint nézi, hogy nyert-e a játékos. E mögött még van a „Backtracking”, ami a tábla generáló algoritmus és „sudoku_tester”, ami a sudoku szabályokat teszteli.

Backtracking: A sudoku egy olyan játék, aminek az a különlegessége, hogy nem lehet matematikailag kiszámolni a megoldását. Ezért ez az algoritmus végigmegy egyesével a cellákon és véletlenszerűen próbálgat bennül számokat, ha a szám jó, akkor továbbmegy egy cellával és ott keres jó értéket, ha rossz értékbe ütközik újra próbál, ha minden lehetőséget kipróbált akkor nincs azzal a kombinációval megoldás, szóval visszalép és az előző cellába új értéket próbál. Így addig megy előre hátra, míg sikeresen le nem generál egy táblát. Viszont ez a programom legnagyobb licitációja is, mivel a nagyobb táblák exponenciálisan több időbe telnek, és

legtöbbször a program kifagyását eredményezik. Mivel mégis ez lenne a különlegessége a programomnak, hogy bármekkora sudokuval lehet játszani, erősen nem ajánlom a 3-asnál (9x9) nagyobbakat. A legnagyobb, amit tesztelésem alatt sikerült generálni az 6-os volt(36x36).

Még egy kellemetlen dolog a Backtracking algoritmussal kapcsolatban, hogy egymásra épülő lineáris számítása miatt nem futtatható több szálon.

Az utolsó játékállapot a sudoku megnyerése után jelenik meg, ami egy toplista, vissza és új játék gomb. A sudoku interfész mielőtt ezt meghívja elmenti a leaderboard.bin fájlba az aktuális menet eredményét. A toplista betölti, szűri és rendezi ezt a toplistát, a szűrés táblaméret, nehézség és segítségek alapján, a rendezés idő szerint növekvő sorrendbe rendezi. A rendező kiválasztásos rendező algoritmust használ.

A savedata.bin és leaderboard.bin nyers adatokat tárol. Mivel nem szöveges és kulcsszavak vagy választókkal vannak elválasztva az adatok nincsen beolvasás tesztelés, azokon kívül, hogy nem lehet olyan nehézséget betölteni, ami meghaladja a megengedettet. A savedata csak 1 adat struktúrát tárol, így nincs szükség semmilyen adattárolási technikára, viszont a leaderboard egy tömb, ezért ennek fájlstruktúrája egy int32 számmal kezdődik, ami megadja az elemek számát, majd a toplista elemek vannak nyersen felsorolva. A toplista elem 12 bájt, és nincs limit, hogy hány ilyen elemet tárolhat a fájl, ezért technikailag akár az int32 maximális értékényi elemet is tárolhat, avagy maximálisan 25.7GB is lehet.

A program memória és cpu használata:
CPU: ~10% (4.2GHZ sebességen 1 logikai processzoron.)
Memória: ~60MB (Nagyobb táblák esetén elérheti a 100MB-t is, viszont ezek a méretek nem kiválaszthatóak az aktuális verzióban.)

A futásidő igény csak pályagenerálásnál releváns, a 4(16x16) méret alatt szinte instant, 4 es és felette játszhatatlanul sok idő.

A programom jelenlegi verziója jelentősen eltér a felkésznél beadottól, mivel időhiány miatt arra kényszerültem, hogy ne a meglévő renderer-t javítsam ki, hanem egy egyszerűbb kirajzolási módhoz folyamodjak.

A tervezett renderer az alábbi módon működött:

- Minden kirajzolandó elem előre kiszámolt, és addig nem változik ameddig rá nem kényszerül.
- Ezeket a kiszámolt elemeket egy tömbbe rakjuk, és a renderer mindent azok paraméterei alapján rajzol ki.

Ezzel a módszerrel azt értem el, hogy lecsökkentettem a képkockák között eltelt időt jelentősen, viszont alapértelmezett struktúra értékek hiánya miatt nagyon hosszú és beláthatatlanná tette az elemek definiálását.

A végső probléma, ami miatt elvetettem, hogy szolid formákból definiáltam a háttereket, ezek soronként kezdő és vég elemekből álltak. Ez jelentősen megnehezítette párhuzamos függőleges vonalak rajzolását, ami mondhatni van jó pár egy sudokuban.

Az új renderer egy igazán pazarló rendszer, avagy semmit nem ment el, mindent újra kiszámol minden egyes képkockába.

Ha lenne időm, ezt a megjelenítési formát én az első módszerrel és design-fájlokban tenném, és több típusra egyedi renderer funkciót adnék be funkció paraméterként, így egy egységes renderer rajzolna ki mindent.

Újra fordítás:

A program újra fordítása egy Makefile-ban valósul meg. A Code::Blocks támogatja a Makefile-t és emlékezetem szerint az egyetemi gépeken egyszer sikeresen lefordítottam vele a programot. De nem vagyok benne biztos, hogy van make a gépen.

Jelenleg csak ez az újra fordítási opció elérhető, mivel több hete működött, leadás elött utolsó héten nem, és akkor nem tudtam rájönni mi a baja. Mivel az, hogy a saját gépemen lefordul nem jelent semmit. Úgy konfiguráltam be Makefile-t használjon.

Makefile Parancsai:

- Debug: compile
- Release: compile
- CleanDebug: Clean
- CleanRelease: Clean // Ezek a codeblocks által hívottak

- all <= Futtatja a programot
- Clean <= Eltakarítja az előző fordított programot újra fordításhoz.
- compile <= Lefordítja a programot.

Az újra fordításhoz szükséges fájlok túl nagyok az InfoC feltöltéshez, ezért a feltöltött állományban van egy szövegfile ami egy github linket tartalmaz.