NHF Kész – Nagy István – Aknakereső

Bevezetés, fontos információk

A programot **CMake** projektként, **GNU/Linux** környezetben, az **ncurses** könyvtár felhasználásával parancssoros kivitelben alkottam meg. A dokumentációt eredetileg Doxygen segítségével akartam megoldani. Ez a forráskódban észre is vehető, viszont jelen dokumentumot kénytelen voltam „kézzel” megalkotni, mivel a Latex alapú pdf dokumentum létrehozását sajnos már nem tudtam megoldani, ugyanis az általam kipróbált Latex disztribúciók mindegyike problémára futott a pdf létrehozása közben, holott a html fájlban minden részlet látható maradt.

Program funkcionalitásai

A program induláskor bekéri a játékhoz szükséges adatokat (játékos neve, pálya mérete, nehézsége). Hiányuk esetén alapértelmezett (statikus) értékekkel indul el.

Létre jön a játék, ami egy aknakereső táblájaként fog megjelenni a felhasználó számára, plusz egy-két információ a játék állásáról (pl. játékidő, zászlók száma, stb.), ebben a felhasználó felfedezhet cellákat, megjelölhet még nem felfedezett cellákat zászlókkal, leveheti a zászlókat. Legalább egy akna szomszéddal, valamint ezek számával megegyező zászlós cella szomszéd esetén a már korábban felfedezett cellára is lehet kattintani, ami ilyenkor az összes szomszédos nem felfedett, nem zászlós szomszédot felfedezi, akár jó volt a zászlózás, akár nem.

A tábla tartalma, valamint a kurzor jelölésén felül statisztikai adatok jelennek meg a kijelzőn (pl. játékidő, zászlók száma, stb.)

A következő bekezdésben a tervezethez képest módosítás történt: Csak egy előző játékállás lett elmentve, ezzel a felhasználóhoz intézett kérdés hamarabbra került a programban. Programozás technikailag ezzel egy fájlban keresést spóroltam meg, bár lehet egy Game listát csináltam volna a fájlból sorozatosan feltöltve, darab számot meg mint a cellák beolvasásánál egy fájl eleji plusz paraméterrel jelölhetném. és betöltésük után csak beemelem az egyező névvel rendelkező játék tartalmát az aktuálisba.

A programból való kilépéskor a játék állapota egy megfelelő fájlba kerül mentésre. Amennyiben ez egy befejezett játék volt, akkor az elért eredmény a Leaderboard.txt állományba kerül, ha viszont egy még folyamatban lévő játékról van szó, akkor annak állapotát a GameState.txt fájlba menti. A következő elinduláskor, ha volt ilyen játékközi mentés, a program megkérdezi a játékost hogy akarja-e folytatni, nemleges válasz esetén a régi játékközi mentés törlésre kerül.

Az aknakereső játékokban nem megszokott, de lehetőség lesz az események visszavonására. Ez egy ki-be kapcsolható funkció lesz, mivel a játék megnyerése, vagy elvesztése teljesen automatikusan befejezik a játékot, míg egy ilyen funkció esetén pont ez a lényeg, hogy egy tévesen felfedezett aknát ki tudjunk például kerülni. Nem egy jó

példa, mivel ilyen funkció aknakeresőkben nem szokott előfordulni, jelenleg is csak programozás technikai kihívás, valamint játéktechnikai oktatásként látom létjogosultságát. Ennek megvalósításához a cellaváltozásokat egy listába regisztrálom, külön megjelölve azokat a cellaváltozásokat, amiket a játékos valamilyen akciója közvetlenül hajtott végre). Visszavonás esetén az utolsó lépéssel megtörtént változások negálódnak, valamint kikerülnek a lista végéről. Visszavonás kezeléses játékok nem kerülnek be a Leaderboard.txt állományába.

A kijelzőn tervezetten csak akkor frissül bármi is, ha a felhasználó valamilyen érvényes inputot ad. Érvénytelen input esetén a program a következő érvényes inputig nem csinál semmit, csak az idő kiírását frissíti *(Új)*.

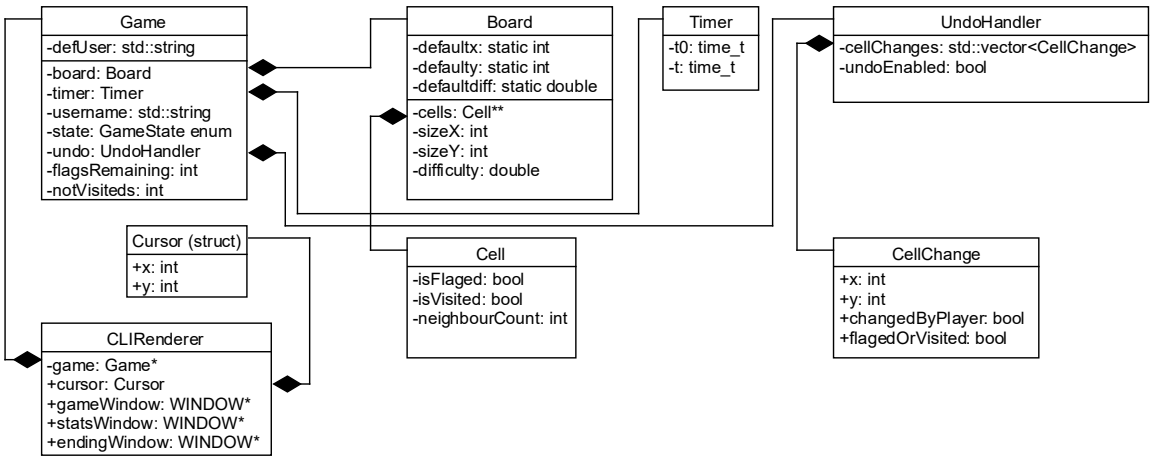
Program használata

Indulás után, ha volt félbehagyott játék, akkor a program megkérdezi, hogy folytatjuk-e. Folytatás esetén betölti a régi játék állást, ellenkező esetben elkezdi az új játék paramétereinek kérdezését a következő sorrendben: Felhasználónév, pálya szélessége és magassága, nehézsége (itt egy 0.0 és 1.0 közötti számot vár, ami az akna cellák arányát jelenti az összes cellához képest), valamint hogy szeretnénk-e a játékban a visszavonás kezelést használni. Visszavonás kezelés esetén a játék nem érhet véget.

A játék folyamatban kurzor nyilak segítségével mozgathatjuk a mutatót, az **F** betűvel zászlózhatunk, a **szóköz** billentyűvel fedezhetünk fel cellákat, valamint a **Q** betűvel léphetünk ki játék közben. Visszavonás esetén a **Z** betűvel lehet visszavonni lépéseket.

Amennyibe az összes nem akna cellát felfedezzük (erről még az ismert hibák részben lesz szó), akkor a játékot megnyertük, és az elért eredmény felkerül egy dicsőség listára.

Osztály leírások



1. ábra Osztályok adat paramétereinek kapcsolata (interfészek nélkül)

A dokumentáció ezen példánya inkább összefoglaló jellegű, a forráskódot doxygen annotációval láttam el, viszont pdf-fet generálni még mindig nem sikerült belőle. Az tartalmazza az interfészeket és minden más belső függvény leírását. A doxygen által generált, html-es verzió megtekinthető a projekt github oldalán (https://github.com/VasutasIsti/AknakeresoCpp)

Cella (Cell)

A pálya legkisebb alkotó eleme, a cella. Cellák alapvetően felfedezetlen, zászlózatlan állapotban jönnek létre, egyes cellák aknákat tartalmaznak. A játék játszását az aknák szomszédai könnyítik meg, ugyanis minden cella kijelzi, hogy hány szomszédja akna. Amennyiben egyik sem, ő egy üres cellának számít, mely felfedezéskor az összes környező szomszédját felfedezi rekurzívan, így megkapjuk a teljes összefüggő üres területet, számozott cellákkal körül véve.

Pálya (Board)

A cellákat fogja össze, valamint a pálya méreteit, illetve a játékos által generált események belső logikáját valósítja meg (pl. összefüggő üres terület felfedése, zászlózás, stb.)

Játék (Game)

A játékot, illetve annak működését figyelő/befolyásoló függvényeket, statisztikai adatokat tartalmazza, legyen az például az eddig eltelt idő, hány zászlót kell még lerakni, avagy ilyen a visszavonás is, valamint az aktuális játékos felhasználóneve. Itt találhatóak azok a metódusok is, amik a játék befejezésekor/elhagyásakor hajtódnak végre.

Cellaváltozás (CellChange)

Cellaváltozás írja le a visszavonáskezelés számára azokat az adatokat, melyből végrehajtható egy adott lépés visszavonása, akár zászlózás, akár felfedezés volt. Utóbbinál lényeges, hogy mivel egyszerre (egy felhasználói lépés után) több cella is felfedeződhet, így le kellett kezelni, hogy egy visszavonás alkalmával az összes ilyen lépés negálódjon.

Visszavonás kezelés (UndoHandler)

Amennyiben engedélyezve van, egy belső listában tárolja a Cellaváltozásokat, és amikor kérés érkezik visszavonásra, a legutolsó változással kezdve azokat elküldi az illetékes osztálynak (Game, aki tovább viszi a megfelelő szintekre).

Időmérő (Timer)

Egy egyszerű időmérő, ami a lekérdezés pillanatában lekérdezi az aktuális időt a ctime segítségével, és egy kezdeti időpillanat különbségével állapítja meg a játék kezdete óta eltelt időt. A ki-be lépések miatt újrakalibrálással is el van látva, így nem lesznek csak úgy maguktól hosszabbak a megjelenített-, mint a tényleges játékidők.

Fájlkezelés

A program a szabályos és tervezett leállása előtt mindenképpen végez valamilyen fájlműveletet, akár egy befejezett játék dicsőséglistára írásáról, akár egy nem befejezett játék teljes állapotának kiírásáról legyen szó. A fájlbaírás és abból beolvasás a stream-re író és arról beolvasó operátorokat hasznosítja, a tesztelés ezt ki is használja.

Megjelenítés

A játék megjelenítéséért a paraméterek bekérését leszámítva az ncurses könyvtár segítségével valósítottam meg, valamint a megjelenítéssel kapcsolatos dolgokat a CLIRenderer osztály fogja össze. Az osztály létrejöttekor három ncurses ablak kerül létrehozásra, egy a játékot meghatározó cellákat jeleníti meg, egy a statisztikai adatokat, illetve egy a játék befejezésekor jelenik meg, jelezve, hogy az adott játékot a felhasználó megnyerte, vagy elvesztette.

Tesztelés

Fontos: Jelenleg van egy main.cpp fájl és van egy test.cpp. Mind a kettőben van main függvény, az itthoni IDE-ben külön targeten vannak, így ez nem okoz problémát. A gtest_lite -os tesztek nagyon szintetikusak, egyelőre még nem tudnak a pályában sokat ellenőrizni, mivel az random generált. Nem biztos, hogy az általam vágyott minden tesztesetre kész lesz a megfelelő ellenőrzés a leadási határidőig (ezen sorok írásakor még az UML NHF-Tervezet -hez képest mért változtatását kéne még befejeznem, mint magasabb rendű probléma), de ezt a github-on külön commitban fogom feltölteni, mint ami a véglegeset tartalmazza.

Utolsó Commit (az eddig a commit-ig feltöltött fájlok készültek el az NHF4 határidejére):

https://github.com/VasutasIsti/AknakeresoCpp/commit/acbb6bb4a397e81c459d6fadc8393288809eaf60

Ismert hibák

Eddig nem tudtam lokalizálni az okát, de az utolsó pár felderítésre váró cella felderítése nélkül is képes véget érni a játék, holott az ezt számontartó számláló papíron nem éri el a küszöbnek meghatározott értéket. Mivel ez nem minden alkalommal történik meg, így ez megmarad bug-nak, TODO: fix later. A játékot nagyon nem befolyásolja, nem omlik tőle össze a program.

Utószó

Sajnos még mindig ördögi csapdában tart az, hogy nem tudok előre tervezni programozásban. Az elméleti órák ebben nem nagyon segítenek ilyen téren. Ha esetleg az olvasó tud ajánlani ezzel kapcsolatban valami ismeretanyagot, szakirodalmat, azt szívesen fogadom. Mint az előző dokumentumokban is írtam, a projekt github-on is megtalálható, a linkje: https://github.com/VasutasIsti/AknakeresoCpp