NHF Skeleton – Nagy István - Aknakereső

Program tervezett 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.

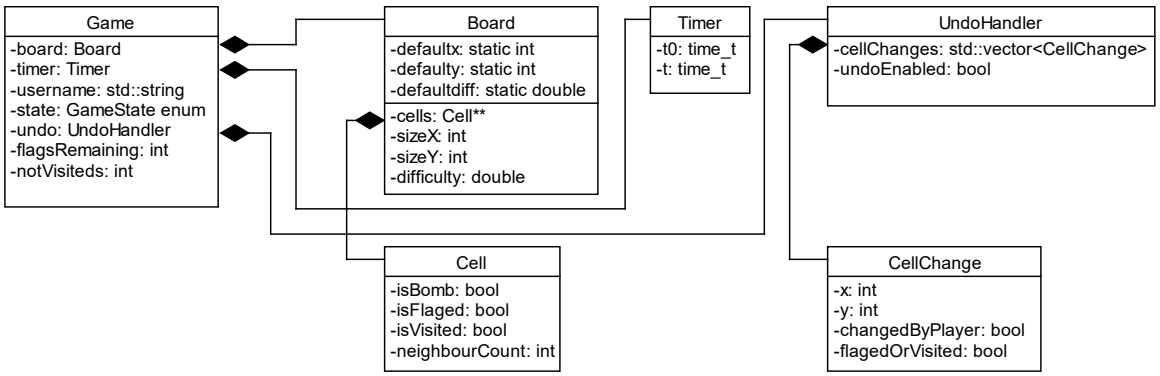
A tábla tartalma, valamint a kurzor jelölésén felül statisztikai adatok jelennek meg a kijelzőn (pl. játékidő, zászlók száma, stb.)

A programból való kilépéskor a játék állapota egy megfelelő fájlba kerül mentésre. Amennyiben ez egy befejezett játék volt, akkor egy, a már befejezett játékokat rögzítő fájlba fűződik a végéhez, ha viszont egy még folyamatban lévő játékról van szó, akkor azt egy külön fájlba írja ki. A következő elinduláskor, ha volt ilyen játékközi mentés, a program megkérdezi a játékos nevet, és ha a játékközi mentésekben van olyan, aminek ugyanaz a játékosa, akkor a program felajánlja, hogy az előzőt töltse be. 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.

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.

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, felfedezett cellára kattintás, 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 erre külön figyelmet kellett fordítani az elkészítés során.

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). (Ezen verzió még a publikus get-ter előttről maradt meg, amennyiben nem kéne külön tesztesetekkel vizsgálnom, nem is biztos, hogy ennyi get-ter függvény lenne, mindenki csak a saját szintjén dolgozna alapon).

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 dokumentáció írásának pillanatában sajnos még nem készült el konkrét fájlba író függvény és azok kezelésére szolgáló logika, viszont stream-re és stream-ről már szinte mindegyik osztály objektuma képes a tartalmát kirakni és arról beállítani magát, a tesztelés ezt ki is használja.

Megjelenítés

Mivel eddig a háttérfolyamatok, valamint a tesztek megírása vitte el az időt, így a megjelenítéssel még nem sikerült foglalatoskodnom. A teszteléshez nem szükséges, viszont a projektől olyan szempontból elkülönül, hogy csak bizonyos részeit használja fel a játéknak, azok belső dolgait közvetlen egyáltalán nem változtatja, csak és kizárólag cellákat fedez fel, valamint zászlókat tesz le, vagy szed fel (Game függvények). A kurzor pozíció is csak a cella egyszerűbb meghatározását teszi lehetővé.

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, egyenlőre még nem tudnak a pályában sokat ellenőrizni, mivel az random generált. Még az sem végleges, meg majd a fájlkezelés befejezésével válik talán véglegessé az automatizáltan tesztelhető dolgok listája.

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