Programozás alapjai 3.Nagyházi dokumentáció

Kovács Róbert Kristóf, R92D9T

Aknakereső

FELHASZNÁLÓI DOKUMENTÁCIÓ

FELÜLET JELLEMZÉSE:

A játék célja, hogy felfedjük a pályán az összes mezőt, amely nem akna. Amennyiben felfedünk egy aknát, a játéknak vége. Segédeszközként használható a jobb klikk a feloldatlan mezőkön. 1 jobb klikkel jelezhetjük, hogy akna van a mezőn, kettővel, ha nem vagyunk biztosak benne. Ezeknek a jelei rend szerint zászló és kérdőjel. Ha harmadszor is jobb klikkelünk a mezőn, eltávolítjuk a jelölőt. Amennyiben a zászló rajta van a mezőn, nem tudjuk feloldani.

Bár induláskor kizárólag a menüt látjuk, ennek leírására később kerítünk sort. A játékbeli felület leírásával kezdjük. 3 részből áll: menüsor, műszerfal, pálya. A pálya mezőkből áll, szinttől függ, hogy hányból. Bal klikkel feloldhatjuk a mezőt. Ekkor a mezőn megjelenik, hogy hány akna van a közvetlen szomszédságában átlókkal együtt. Ha nulla, akkor a mező nem ír ki semmit és a szomszédjai is felnyílnak egészen addig, míg a feloldott területet nem számok, vagy pályaszélek határolják. Ha feloldunk egy aknát, akkor a mezőn felrobbanó aknát látunk és a játék véget ér. Ekkor a játék felfedi a többi aknát is, hogy láthassuk, merre voltak.

A műszerfal többletfunkciókat ad a játékhoz. A műszerfal bal oldalon a zászlószámláló található, ami eredetileg a pályán elhelyezett aknák számával egyenlő. De a letehető zászlók száma nem korlátozott annyi kikötéssel, hogy egy pályabéli mező csak egy zászlóval jelölhető meg. A számlálót tehát negatívba is vihetjük. Zászlót úgy tehetünk le, ha egy üres és felfedetlen pályamezőre jobb klikkelünk. A zászló használata abban segít, hogy megjelöljük azokat a mezőket, amelyekről úgy véljük, hogy aknát tartalmaznak. Ha zászló van egy mezőn, akkor nem fedhető fel. Ha jobb klikkelünk egy zászlóra, a mezőn kérdőjel jelenik meg. Ekkor már feloldhatóvá válik. Ha egy kérdőjelre kattintunk jobb egérgombbal, akkor üressé válik a mező. Ha eltüntetünk egy zászlót a zászlószámláló növekszik egyel.

Középen a smiley gomb van. Alapvetően mosolyog. Amíg nyomva tartjuk az egeret egy mezőn, aggódik, ha aknát fedünk fel halottá válik, ha megnyerjük a játékot, napszemüveget kap és mosolyog. Lekattintása újraindítja a játékot az előző módban.

A jobb oldalon van a számláló. Az első mező felfedésétől a játék végéig számol. Ha megnyitjuk a menüsor bármelyik elemét, a számláló megáll a számlálásban, visszalépve folytatja.

A MENÜSORRÓL RÉSZLETESEN:

A menüről részletesen:

- New game: ugyanúgy új játékot indít, mint a smiley gomb.
- Load Game: megadott elérési útú és nevű fájlt visszatölthetünk és folytathatjuk a játékot. Ekkor a fájl törlődik, hogy ne tudjuk biztonsági mentésnek használni, ami csalás lenne.
- Mode: Külön menü melyben a következők állíthatók be:
    - Beginner: kezdő nehézségű játék indítása.
    - Intermediate: haladó nehézségű játék indítása o Expert: szakértő nehézségi szintű játék indítása.
    - Custom: Egyedi szint. Ekkor egymás után megjelenő mezőkben adhatjuk meg a pálya adatait:
        - A pálya szélessége legalább 9, maximum 24.
        - A pálya magassága legalább 9, legfeljebb 30.
        - A pályán elszórható aknák száma 10-300-ig terjed, de a pályán lévő mezők számától 1-el kevesebbnek kell lennie. ▪ Enter megnyomásával vagy a „tovább” gomb használatával tudjuk elindítani a játékot. Ekkor a menü eltűnik és a játék indul.
- A Leaderboard pontban megnézhetjük az időeredményeket. Ha Anonymous 999 jelenik meg akkor az gyári adat. Még nem írta felül játékos. A kilépéshez a „vissza” gombra kattinthatunk.
- Az utolsó a kilépés. Ez bezárja a programot.

A második menüsor a pause:

- Continue: folytatja a játékot az előző pozícióból.
- Save Game: egy általunk kiválasztott megadott nevű és helyű új fájlba menthetjük az aktuális állását a játéknak, majd a menübe dob. Ezt egy külön ablak segíti.
- Help: Ekkor a játék alapvető rövid leírása és a fejlesztéssel kapcsolatos minimális információ látható, egy külön információs ablakban. A kilépéshez kattintson a „OK” bombra.

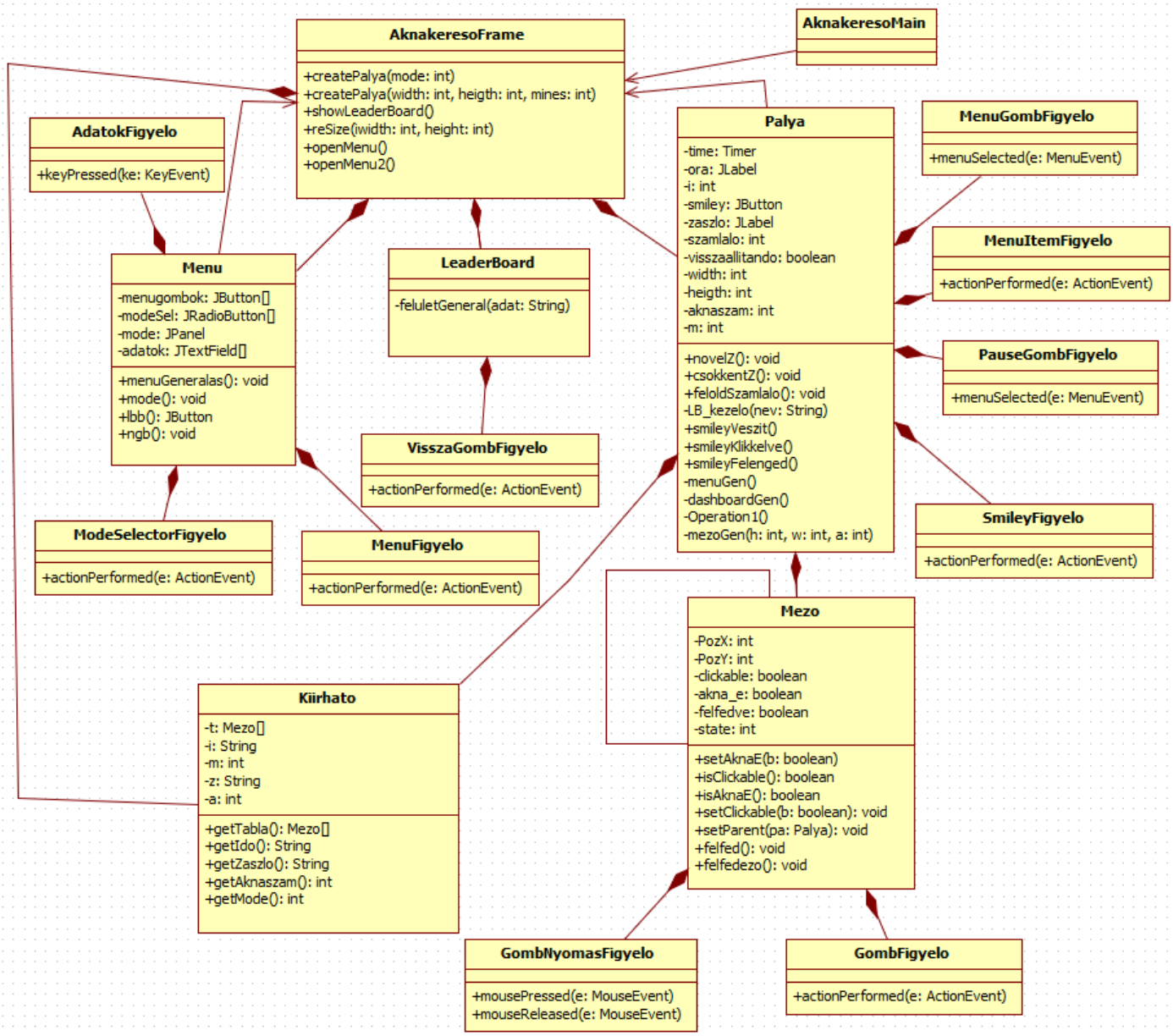
A menübe lépés minden esetben a játék befejezésével jár. Azaz visszalépésre nincs lehetőség, új pálya vár minket, de a Pause gombra kattintva csak megáll a számláló és később folytathatjuk. Ez utóbbi nem teljes felület, mindössze lenyíló menü, de a játékteret elfedjük, hogy ne lehessen előnyhöz jutni.

A DICSŐSÉGTÁBLÁRÓL:

Ha megnyerjük a játékot, és az eredményünk kisebb, mint az előző legjobb 3 valamelyike, akkor bekerülünk a dicsőség listába. Ekkor a program bekéri a nevünket. A dicsőségtábla háttérfájljába ez bekerül az időeredményünkkel és nehézségi szinttel, de a dicsőségtáblában is rögtön látható lesz. A tábla csak a legjobb 3 eredményt rögzíti szintenként. Az egyedi szint nem tartozik bele ebbe a listába, kizárólag a szabványos pályák.

PROGRAMOZÓI DOKUMENTÁCIÓ

UML DIAGRAM



A program működése:

A AknakeresoMain legenerál egy AknakeresoFrame-et. A frame kreál egy menüt, amit megjelenít és a menü a Framnek jelzi a felhasználó által választott lehetőséget egy függvényhívással. Ugyanígy a pálya és leaderboard is a frame-mel kommunikál. A Palya osztály Mezői egymással és a Pályával is kommunikálnak.

AKNAKERESOFRAME:

`createPalya(int mode)`: Szabványos pálya generálását végzi és megjeleníti, a menüt elrejti

`createPalya( int width, int height, int mines)`: egyedi pályát lehet vele generálni és megjeleníteni.

`loadPalya(File f)`: egy megadott fájlból lehet betölteni, generálni és megjeleníteni a pályát. A megadott filet ki is törli.

`showLeaderBoard()`: minden játék után változhat a leaderboard, így minden alkalommal érdemes újra legenerálni. Meg is jeleníti a képernyőn.

`reSize(int width, int height)`: A megjelenített osztályok beállíthatják a frame méretét igényeik szerint.

`openMenu()`: pályáról menübe lépést végzi.

`openMenu2()`: a dicsőségtáblából a menübe léptet.

`Menu getM()`: tesztelés céljából a menü lekérhető

`LeaderBoard getL()`: tesztelés céljából az aktuális leaderboard lekérhető

`Palya getP()`: tesztelés céljából az aktuális pálya lekérhető

A MENU:

`menuGeneralas()`: legenerálja a menü felületét.

`mode()`: a módválasztó panelt generálja le.

`lbb()`: A dicsőségtábla megnyitását segítő gombot lehet lekérni teszteléshez.

`ngb()`: Az új játék indítását segítő gomb lekérhető teszteléshez.

Eseménykezelők:

`MenuFigyelo:`

- **New game gomb esetén:**
  Megállapítja, hogy mi a kiválasztott mód a a rádiógombok közül és meghívja a createPalya() függvényét az AknakeresoFrame-nek egy kódra. Amennyiben egyedi

pályát választunk, a mezőkből kiolvassuk az adatokat és amennyiben túl van valamelyik a szabály szerinti határértéken, akkor módosítjuk őket, majd a createPalya() felüldefiniált változatát hívjuk meg.

- **Load Game:**
  Egy showOpenDialog() segítségével bekérjük az elérési útvonalat és a kapott File példánnyal paraméterezve a loadPalya() függényt hívjuk a framen belül.
- **LeaderBoard:**
  a frame showleaderboard függvényét hívja.
- **Exit:**
  bezárja az alkalmazást

`AdatokFigyelo:`

Azt figyeli, hogy az egyedi pálya JTextField mezőibe csak szám karaktert lehessen bevinni.

`ModeSelectorFigyelo:`

Aszerint, hogy az egyedi pálya mód van-e kiválasztva vagy sem, a hozzátartozó beviteli mezők elérhetőségét állítja.

LEADERBOARD:

`feluletGeneral(String[] adat):` adattömböt kap, amelyből legenerálja a felületet. A beolvasás a konstruktor feladata.
`JButton backB():` tesztelés céljából legenerálható a felület.
Eseménykezelő:

`VisszaGombFigyelo:`
A frame openMenu() függvényét hívja.

PALYA:

`novelZ():` növeli a zászlószámláló (zaszlo) JTextField-be írt értéket.

`csokkentZ():` csökkenti a zászlószámláló (zaszlo) JTextField-be írt értéket.

`feloldSzamlalo():` A szamlalo nevű változót növeli, ha egy mező meghívja. Ha a szamlalo = összes mező – aknás mezők: akkor a kattinthatóságát kikapcsoljuk a mezőknek. A Smiley gomb megkapja a győztes textúrát (t17), majd a számlálót leállítjuk. A felhasználónak

bekérjük a nevét egy `showInputDialog()` segítségével. Majd az `LB_kezelo()` függvényt hívjuk a kapott névre.

`LB_kezelo(String name):` Ha a kapott név hosszabb 15 karakternél, akkor csonkolunk. Ezután beolvassuk a táblát. A mode változóból számolva megállapítjuk, melyik 3 elemre kell tesztelni és egy ciklussal végig megyünk rajtuk. Amennyiben találunk egy értéket, ahova betehetjük az új elemet (kisebb az időérték), akkor lejebb toljuk az alatta lévő elemeket (utolsó kiesik) és beszúrjuk méltó helyére az új rekordot. Ezután visszaírjuk a táblát a fájlba.

`smileyVeszit():` Megkapja a smiley.

`smileyKlikkelve():` Megkapja a smiley a lenyomás textúrát.

`smileyFelenged():` Ha felengedjük a smiley-t visszaállítja, amennyiben a játék közben nem lett elveszítve/megnyerve.

`menuGen():` Legenerálja a menüsort.

`dashboardGen():` Legenerálja a műszerfalat.

`mezoGen(int h, int w, int a):` Legenerálja a pálya mezőit. Majd random sorsolt pozíciókat aknává alakít a mező a `setAknaE()` függvényével.

Tesztelés céljából továbbá létezik: `MM(), PM(), getMezo(int s, int o), getS(), getIdo().` Feladatuk rendre a menübe lépés a Pause menü gombjainak egy mező lekérése pozíció szerint, és idő lekérése intként.

Eseménykezelők:

`SmileyFigyelo:`

Újraindítja a játékot. Törli a régi mezőket, visszaállítja a pálya változóit és a műszerfalat. Új mezőket generál, majd elindítja a Timert és frissíti a felületet.

`MenuGombFigyelo:`

A frame openMenu() függvényét hívja.

`PauseGombFigyelo:`

Elrejti a mezőket és megállítja a timert. Eközben a JMenuItem-ek megjelennek.

`MenuItemFigyelo:`

- **Continue:**
  Elindítja a számlálót és láthatóvá teszi a mezőket.
- **Save Game**
  A mezők parentjét null-ra állítjuk, hogy maga a pálya ne kerüljön kiírásra, ezután létrehozunk a pálya adataira és mezőire egy Kiirhato osztálypéldányt, amelyet kiírunk egy showOpenDialog() által kiválasztott fájlba.
- **Help**
  showMessageDialog() jelenik meg rajta némi szöveggel.

MEZO:

`isClickable():` visszakült egy booleant, miszerint reagál a kattintásra, vagy sem.

`setAknaE(boolean b):` Beállítható, hogy akna legyen-e a mező

`isAknaE():` Visszakapjuk, hogy a mező jelenleg akna-e.

`setClickable(boolean b):` Beállítható, hogy a gomb reagáljon e a kattintásra.

`setParent(Palya pa):` Beállítható a tartalmazó pálya. Ez a függvény hozzáadja az eseménykezelőket is.

`felfed():` Ha a gomb kattintható és nem zászló fedi, valamint nincs felfedve már, akkor összes szomszédját megkérdezve `(isAknaE())` kiszámoljuk, hogy hány akna van a szomszédjában és eszerint kap a Mezo új textúrát. Amennyiben 0 akna van a szomszédságában, meghívja a `felfedezo()`-t. Továbbá a vezérlést kezelő változók értékeiről is gondoskodik.

`felfedezo():` Az összes szomszédos mező felfed() függvényét hívja, ami így egy rekurzív algoritmust eredményez.

Eseménykezelők:

`GombFigyelo:` Ha akna, akkor kattinthatatlaná teszünk mindent, felfedünk minden mezőt, ami akna és a smileyVeszit() függvényt hívjuk. Ellenkező esetben csak felfed() hívása történik.

`GombNyomasFigyelo:` Kezeli a jobb klikkes jelölőket és a lenyomásra történő smiley gomb változást vezérli.

KIIRHATO:

Tárolja a következő információkat a pályáról: Mező mátrix, idő és zászló stringként, aknaszám és mód.

PAINTEDPANEL:

A sima JPanel átszínezett változata.

AKNAKERESOMAIN:

Ez alkot és hív meg egy AknakeresoFrame-et. Ha bezáródik a frame. A program is bezáródik.