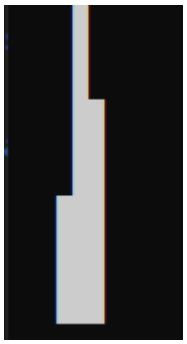
Stack to the top – dokumentáció

Felhasználói kézikönyv

A program kizárólag a rövidtávú unaloműzésre ad megoldást. Konzolos körülmények között játszhatja a felhasználó a „Stack To The Top” nevű játékot.

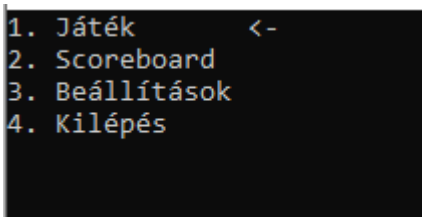
A játék célja, hogy a pálya aljától annak plafonjáig „építkezzen” a rendelkezésre álló bábuval. Hiba lehetőség, ha a bábuk nem illeszkednek megfelelően. A rosszul illeszkedő elem, lelógó darabja leesik, és arra már nem építhetünk.



Illusztrációs példa egy teljesített pályáról.

A felhasználónak a játék indítását követően, egy gombbal kell a játékot vezérelnie. „SPACE” gomb megnyomását követően, rögzül az adott pontban a bábu.

Menü



A menürendszer a korábbi évtizedek mobiltelefonos játékait idézi. A billentyűzet „nyíl” gombjaival tudunk navigálni benne. Egy „<-” ikon jelöli az aktív menüpontot. A megfelelő menüpont kiválasztása után „Enter” billentyűvel léphetünk be abba.

Menüpontok funkciói:

- Játék: Ez a menüpont indítja a játékot magát.
- Scoreboard: A már elmentett eredményeit a felhasználó itt találja meg. A játék elmenti a játékos által megadott „nicknevet”, a teljesítés pontos időpontját, és a nehézségi fokozatot.
- Beállítások: Ezen a menüponton belül kizárólag a játék nehézségi szintje állítható be. Az előre definiált három nehézségi szint a bábu mozgási sebességére van kihatással.
- Kilépés: Ennek kiválasztásával zárhatjuk be a szoftvert.

Fejlesztői dokumentáció

A fejlesztés első lépése az alapvető játékmenet lemodellezése volt, kódolással, hogy a megoldandó probléma jól átlátható legyen.

Ekkor fogalmazódott meg, hogy a pályát egy 10x10-es méretű kétdimenziós tömb határozza meg. A tömb elemeinek alapvető értéke legyen nulla, és később egyesek fogják jelölni azt, ahol a pályán már van elem, vagy az éppen aktuális hova fog kerülni.

A pálya és annak működési elvének definiálása után a képernyőn történő megjelenítés volt a következő cél. Egy tíz sor magasságú tömb az esetükben azt jelenti, hogy a felhalmozandó bábu „torony”, tíz sor magas lesz, így érdemes nyitni egy számláló ciklust, ami a pálya magasságával egyenlő értékig fut, a ciklus magjában pedig az adott sorban történő folyamatok algoritmizálása fog történni. Konkrétan a feladat a ciklusmagban: Egészen addig ameddig a felhasználó nem nyom „space” gombot, addig a bábu folyamatosan mozogjon jobbra, amíg eléri a pálya szélét. Az elérést követően induljon el újra a pálya másik szélétől. A „space gomb megnyomását követően”, rögzítsük a pályát képző tömbben az az aktuális sorunk adatait, majd a számláló ciklusunk léptet minket a következő sorba.

Tehát az eddigi megoldás:

```
Palya[,] = int[10,10]
Palyafeltoltes_Nullakkal()

babu_mozga(){
        Ciklus i = 0 tól, 10-ig
                y = oszlop = 1
                Ciklus amíg Felhasznlo nem nyom "space"-t Addig:
                        x = sor;
                        Ha (y < 7) y++
                        Különben y = 0
                        babu_kirjazol(x, y, y+1, y+2)
                Ciklus vége
                rogiztes_a_palyan()
        Ciklus vege
}
```

Az eddigi megoldásra tekintettel a következő megoldandó problémák merülnek fel:

- Kirajzolásnál az egész eddigi pályát kikellene rajzolni. A megoldása egyszerű, a kirajzoló függvényen belül ciklussal átkell futni a pályán, és ahol egyet érték van a cursort oda helyezve, ki kell írtani a karakter.

- Ha folyamatosan egymásra rajzol a program, az hamar átláthatatlan lesz, és egyértelműen nem megfelelő a megoldás szempontjából. A megoldás: a kirajzolás előtt, tüntessük el az egész addigi tartalmat. (c#-konzol : Console.Clear();)
- Legyen némi időköz a kirajzolások között, hogy legye ideje a felhasználónak a reagálásra. A megoldás a konzol fagyasztása. (c#-konzol: Thread.Sleep();)
- A rögzítés egyszerű, a rögzítő függvény megkapja, hogy melyik sorban és, hogy melyik oszlopban jár a bábu, a tömb azon sorát, és oszlopait(függően a bábu aktuális hosszától) egyesre állítja. Viszont, hogy ne lehessen akárhova pakolni, csak és kizárólag olyan helyre, ami alatt van már bábu, ellenőrizni kell, hogy az előzőleg rögzített sornak azon oszlopaiban, amelyekben az aktuális sorunkban rögzíteni kívánunk, egyes van-e. Ha nem egyes van, akkor az aktuális sorban, a hibás oszlopban ne rögzítsen!

```
rogiztes_a_palyan(sor, oszlop){
        Ciklus i = 0-tól Bábu_hosszáig:
                Ha palya[sor-1,oszlop+i] = 1 Akkor palya[sor,oszlop] = 1
                Különben Bábu_hossza--
        Ciklus vége
}
```

Ezzel a játékszoftver magja elméletben megszületett, a következő a menü rendszer, és a játék további hangolásának kitalálása.

Az a fejlesztés elején biztos volt, hogy szeretnénk egy Scoreboard menüt, ahol a statisztikák vannak, és egy beállítások menüt, ahol a felhasználó módosíthat a „nehézségen”, ezzel interaktívabbá téve a játékot.

Menü működése

Az ismert menüpontok tekintetében, létrehoztunk egy tömböt, amiben négy érték található. Alapból így néz ki: {1, 0, 0, 0}

Ez a tömb jelképezi azt, hogy éppen melyik menüpont lesz a kiválasztott a felhasználónál. Ide is fog kerülni „<-” ikon.

Következő lépésként kikell íratni a konzolra a menü pontokat, és a nyíl ikont az éppen aktív pontra. Ezt egy végtelenített ciklussal oldottuk meg, ami folyamatosan törli a konzol tartalmát, majd kiírja őket újra, a nyíllal együtt.

Innen jön a következő lépés, hogy a felhasználó interakcióba tudjon lépni a menüvel. Mielőtt a végez a végtelen ciklus az aktuális magjával, vár egy karaktert a felhasználótól. A karakter leütése után ellenőriznie kell a programnak, hogy az milyen karakter volt. Három lehetőséget kezel a szoftver: fel-le nyilak, és az enter. Ha előbbi kettő közül érkezett a karakter, akkor azt jelenti, hogy navigálni

szeretnénk a menüpontok között, ha „enter” a beérkezett billentyű, akkor megnyitni az adott pontot.
Nézzük először a navigálást!

Ha fel vagy le gombot észlel a program, akkor mind a két esetben ugyan az a függvény fut le, csak más attribútummal. Felfele mutató nyíl esetén ez egyes, lefele mutató nyíl esetén ez 0.

Ezután a függvény a menü tömbben megnézi, hogy melyik az aktív menüpont, és a paraméterétől függően, átállítja a másik értéket egyre, az előzőt pedig nullára. Pl: Eredeti állás: (0,1,0,0); Lefele nyíl -> Módosított állás: (0,0,1,0)
Ez után végzett a menüt kiíró ciklus, és indul újra. Törli az előző tartalmat, és kiírj az újat, a „<-” ikon új pozíciójával.

A másik alternatíva az „Enter” volt, ekkor megnyitja az adott menü pontot.
A működése egyszerű, megnézi a menüt jelképző tömböt, hogy melyik az aktív elem, és annak ismeretében dönti el, hogy melyik legyen a következő függvény, amit megnyit. Ha az első pont az aktív, akkor a játékot indítja, ha a második akkor a Scoreboard függvénye indul el, a harmadik menüpont esetében a beállítások függvény fut le, viszont, ha negyedik menüpont aktív, akkor egy alapvető c# függvényt hív meg az „Enviroment.Exit”, és kilép a program.

Scoreboard menüpont

A függvény megnézi, hogy létezik-e a „saves.txt”, ha nem létezik, akkor egy üzenettel közli azt a felhasználó részére. Ha létezik, akkor a tartalmát kiolvasva, szimplán kiírja azt a konzolra. A tartalomra példa: „Kornél 2020.12.24 Könnyű fokozat.”
Tabulátorral elválasztva tárolja a játékost, aki teljesítette a pályát, a dátumot, hogy mikor, és, hogy milyen fokozaton teljesítette.

Beállítások

Ez is egy egyszerű függvény. Kiírás formájában tájékoztatja a felhasználót a lehetőségeiről, majd várja tőle, hogy szám formájában megadja a nehézségi szintet. (1-3)
A bevitelt követően ellenőrzi a program, hogy megfelel-e a kritériumoknak, ha igen, akkor állítja a „Difficulty” változó értékét, a felhasználó által bevitt értékre.

A megtervezést követően már csak a pontos kód sorok megírása, és egybe fésülése volt hátra. Az így keletkező esetleges hibák javítása, és a program játszhatóvá tétele olyan rövid kódsorokkal, ellenőrzésekkel, amik elengedhetetlenek.

Min PL.:

- A játékmenet magját képző 10-ig számláló for ciklusba ellenőrzés gyanánt figyelni a karakterünk hosszát, ami alapból három, de a hibák során fogy. Ha eléri a nullát breakelje a ciklust, a játék kilépjen a menübe, egy üzenettel kísérve. Ha a ciklus sikeresen végig fut, és a karakter hossza nagyobb, mint nulla, akkor a felhasználó lehetőséget kapjon magát elmenteni a „saves.txt”-ben, egy fantázianév megadásával.

A szoftvert készítette Fél Ferenc és Hencz Kornél Tibor
Szoftvertervezés és fejlesztés beadandó feladat