Óbudai Egyetem

Programozás I. féléves beadandó

SSUD (Simple Single-User-Dungeon)

Novák Adrienn

Laborvezető:
Szántó Balázs

Neumann János Informatikai Kar

Budapest, 2012. november 24.

Tartalomjegyzék

1. Feladat

SSUD (Simple Single-User-Dungeon)
(http://en.wikipedia.org/wiki/Roguelike, http://en.wikipedia.org/wiki/Nethack)

Készítsen el egy programot, amely valamilyen módon (pl. előre definiált 2D tömbben) tárol egy egyszerű, egy képernyőn elférő (vagyis: maximum 80x25 méretű) labirintust. A program elindulása után jelenítse meg a labirintust (pl. fal=piros # karakter), ezután a felhasználónak legyen lehetősége a labirintusban a saját karakterét (zöld @ karakter) a labirintusban mozgatni (kurzor-gombokkal történő mozgatás: ld. lent). A játékot jelenítse meg a konzol 80x25-ös képernyőjén megfelelő konzol-beállító tulajdonságok (Console.Top, Console.Left, Console.ForegroundColor) segítségével. (1. ábra)

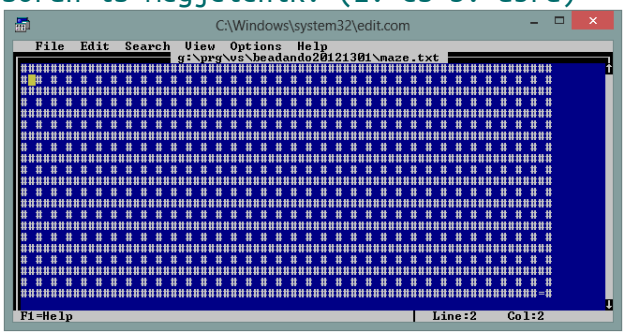
```
ConsoleKeyInfo x = Console.ReadKey(true);
if (x.Key == ConsoleKey.LeftArrow) Console.CursorLeft--;
```

1. ábra

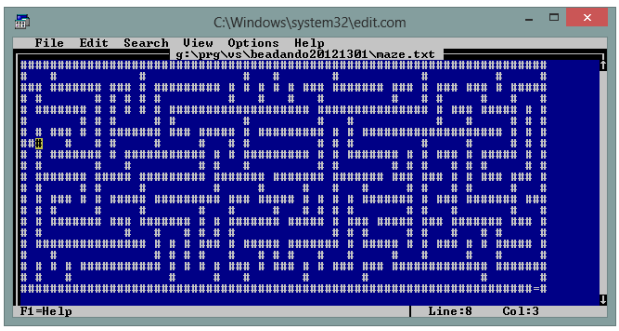
2. Megvalósítás

A játékot idő szűkében nem algoritmus által generált, hanem előre „megrajzolt” labirintusokban lehet játszani, a játék elején ezek körül választ a gép egyet. Hogy ne legyen annyira egyhangú, a felhasználó által megadott számú csillag is elhelyezésre kerül a pályán, kijutni ezek összeszedegetését követően lehet. A felhasználó játék közben láthatja, hogy hány lépést tett meg, hogy épp melyik pályán játszik, illetve hogy hány darab csillagot kell még összeszednie, mielőtt kinyílna az ajtó.

Egy kis érdekesség, hogy a pályákat a Windows saját beépített konzolos szövegszerkesztőjével (edit.com) készítettem, mivel ebben tudtam valóban olyannak látni a rajzolás alatt álló pályát, ahogyan az a végleges használat során is megjelenik. (2. és 3. ábra)

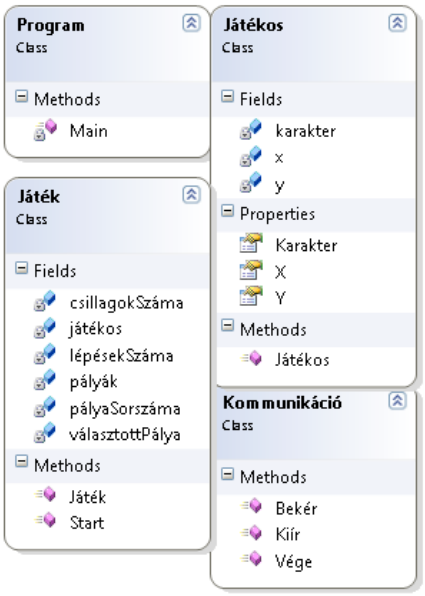


2. ábra



3. ábra

2.1 A megvalósítás során alkalmazott osztályok



4. ábra

A feladat megvalósításához 3 osztályt használok, az egyik a Játékos osztály, amely magát a felhasználót hivatott jelképezni az útvesztőben. A Játék osztályra hárul a munka legnagyobb része – a játékmező kiválasztása és előkészítése is itt történik. A Kommunikáció osztály egy viszonylag gyenge kísérlet a megjelenítés és az „üzleti logika” elkülönítésére. (4. ábra)

2.1.1 Játékos osztály

Az osztály mezőinek áttekintése:

Az ***x*** és ***y*** mezők a játékos éppen aktuális pozícióját tartják számon, a ***karakter*** mező pedig a képernyőn a játékost jelképező ('@') karaktert tárolja.

Az osztály tulajdonságainak áttekintése:

A mezőkhöz tartozó publikus, a karakter esetében csak olvasható, a pozíció esetében pedig írható és olvasható tartalmú változók.

Az osztály metódusainak áttekintése:

Az osztály egyetlen metódusa egy paraméter nélküli konstruktor, a játékos minden esetben a bal felső sarokban kezd.

2.1.2 Játék osztály

Az osztály mezőinek áttekintése:

csillagokSzáma: a játék egyetlen felhasználó által megadott paramétere, a pályán elhelyezésre kerülő, majd később a még össze nem gyűjtött csillagok számát tárolja.
játékos: egy Játékos osztály egy példánya
lépésekSzáma: a játék mintegy érdekességképpen számon tartja, hogy egy adott játék alatt hány lépést tett meg a felhasználó.
pályák: a játékhoz használható labirintusok háromdimenziós karaktertömbje.
pályaSorszáma: a játékhoz véletlenszerűen generált szám, a 3 dimenziós tömb egy 2 dimenziós részének indexe.
választottPálya: a kiválasztott pálya megjelenítéséhez szükséges karakterek 2 dimenziós tömbje. Ebbe kerülnek a csillagok is.

Az osztály tulajdonságainak áttekintése:

Nincsenek. A játék példányosítást követően a Start metódus hívásával elindul.

Az osztály metódusainak áttekintése:

Játék: Konstruktor. A véletlenszerűen kiválasztott pályán elhelyezi a felhasználó által megadott mennyiségű csillagot.
Start: A játék lelke, gyakorlatilag egyetlen végtelen ciklus, amely addig fogad billentyű lenyomásokat, amíg a felhasználó az összes csillagot össze nem szedi és nem mozgatja a karakterét a jobb alsó sarokban található '=' karakterre.

2.1.3 Kommunikáció osztály

Minden beolvasás és kiíratás ennek az osztálynak a segítségével történik.

Az osztály metódusainak áttekintése:

Bekér: A kezdő képernyő megjelenítése és a csillagok számának bekérése történik itt, egy játék alatt egyetlenegyszer fut le.

Kiír: A képernyő törlése, majd a teljes játékmező (újra)megjelenítése (a labirintus, a játékos és a csillagok), valamint az aktuális pálya sorszáma, a még össze nem gyűjtött csillagok, illetve a megtett lépések számának kiírása is ennek a metódusnak a lefutása következtében történik meg. Egészen addig fut le újra és újra, amíg a játékos el nem éri a célját.

2.2 Alkalmazott fontosabb algoritmusok ismertetése

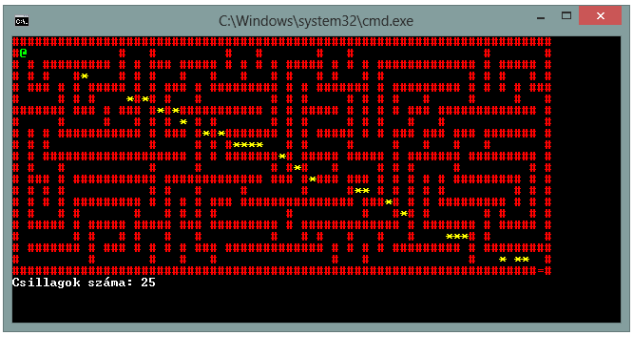
2.2.1 Játék vezérlése

A nyílbilletnyűk segítségével. A játékos akkor léphet adott irányba, ha ott nincs fal. ('#' karakter) A csillagokat rajtuk történő áthaladással lehet összeszedni, a játék végén a játékos karaktert az „ajtóra” ('=' karakter) mozgatva lehet befejezni a játékot.

2.2.2 Játékmező előállítása

Az egyetlen említésre méltó rész a kódban a csillagok véletlenszerű elhelyezése a táblán.

Először két random számmal próbálkoztam, de az eredmény a várttól eltérően túl szabályosra sikerült: annak ellenére, hogy a pálya nem négyzet alakú, a csillagok, mintha vonalzóval rajzoltam volna a helyüket, egytől-egyig a pálya átlójának vonalában foglaltak helyet. (5. és 6. ábra)



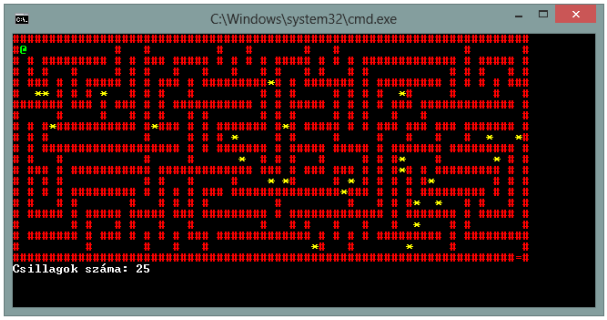
5. ábra

```
Random rndX = new Random();
Random rndY = new Random();
while (cs > 0)
{
    int x = rndX.Next(1, 71);
    int y = rndY.Next(1, 21);
    if (választottPálya[y, x] == ' ')
    {
        választottPálya[y, x] = '*';
        cs--;
    }

}
```

6. ábra

Következő lépésként úgy döntöttem, egy harmadik random számot is „kérek”, majd ezt adom meg seedként a koordináták előállításához. Az eredmény jobb lett, de még mindig nem éreztem tökéletesnek. (7. és 8. ábra)

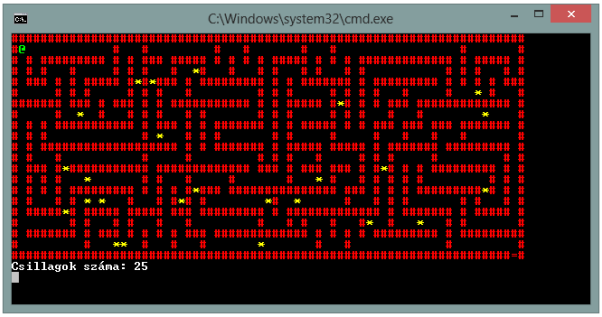


7. ábra

```
Random rndSeed = new Random();
Random rndX = new Random(rndSeed.Next(1, 1000));
Random rndY = new Random(rndSeed.Next(1, 1000));
while (cs > 0)
{
    int x = rndX.Next(2, 71);
    int y = rndY.Next(2, 21);
    if (választottPálya[y, x] == ' ')
    {
        választottPálya[y, x] = '*';
        cs--;
    }
}
```

8. ábra

Harmadik nekifutásra már végre elfogadhatónak tűnt az eredmény. (9. és 10. ábra)



9. ábra

```
Random rndSeed = new Random();
Random rndX = new Random(rndSeed.Next(1, 1111));
Random rndY = new Random(rndSeed.Next(2, 2222));
while (cs > 0)
{
    int x = rndX.Next(2, 71);
    int y = rndY.Next(2, 21);
    if (választottPálya[y, x] == ' ')
    {
        választottPálya[y, x] = '*';
        cs--;
    }
}
```

10. ábra

2.3 Specifikáció

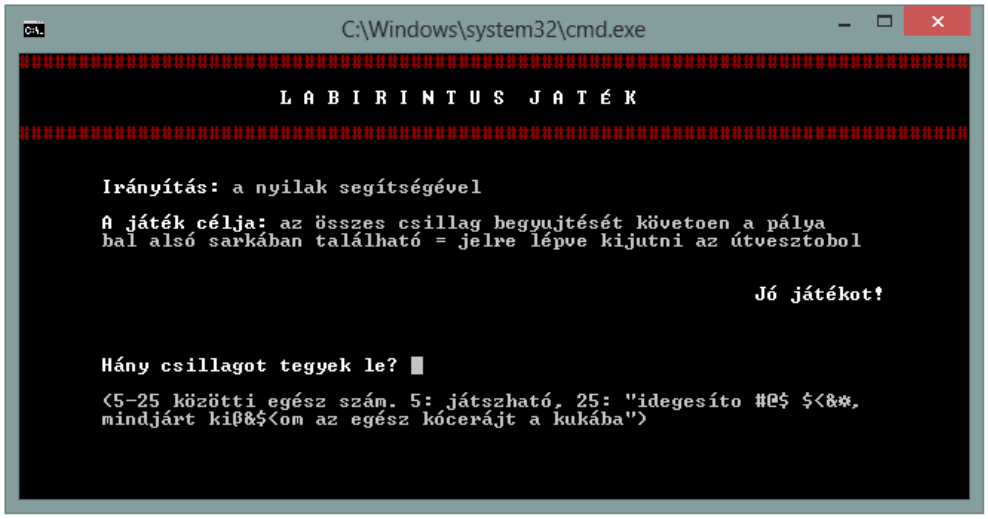
A játék elején a program egy egész számot vár bemenetként, 5-nél kisebb szám esetén a csillagok száma 5, 25-nél nagyobb szám illetve nem megfelelő típusú bemenet esetén 25 lesz. A labirintus kiválasztása véletlenszerű.

A játékos a nyílbillentyűk segítségével mozog a játékmezőn, a cél a csillagok összeszedegetése, majd az útvesztőből való kijutás.

A játék akkor ér véget, ha az összes csillag begyűjtésre került és a játékos az '=' karakterre lép.

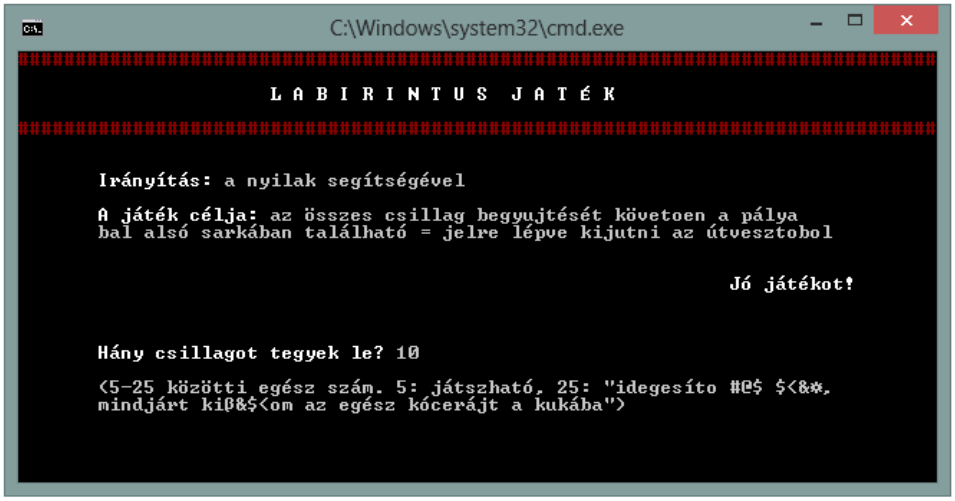
3. Felhasználói segédlet

A játék kezelése pofonegyszerű: a nyitóképernyő már önmagában is kellő mennyiségű információt biztosít. (11. ábra)



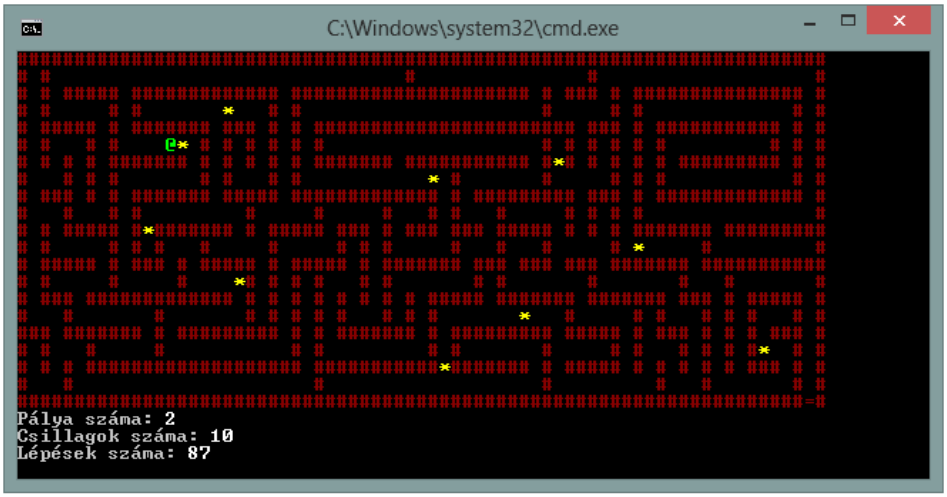
11.ábra

A csillagok száma 5 és 25 között lehet, ettől eltérő esetekben 5-nél kisebb szám megadása esetén 5, 25-nél nagyobb szám illetve bármi más (pl. betű, „üres Enter”) megadásakor 25 csillag kerül a játékmezőre. (12. ábra)

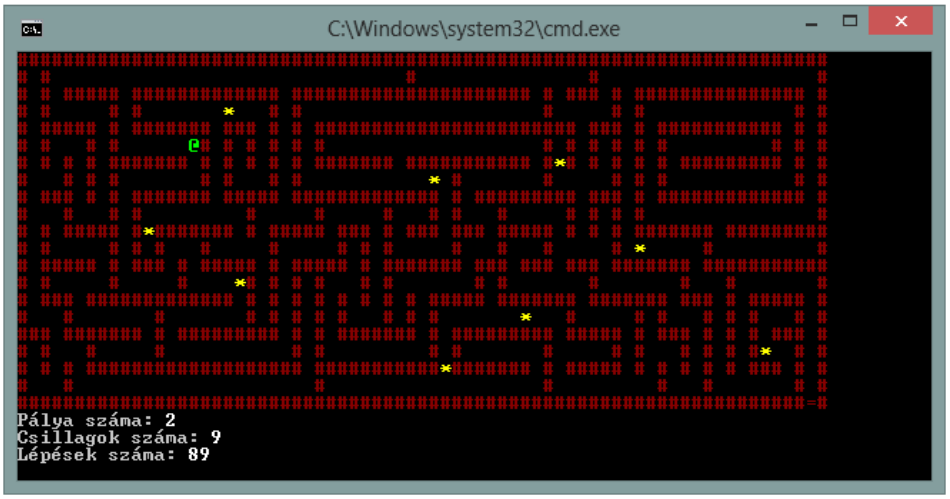


12. ábra

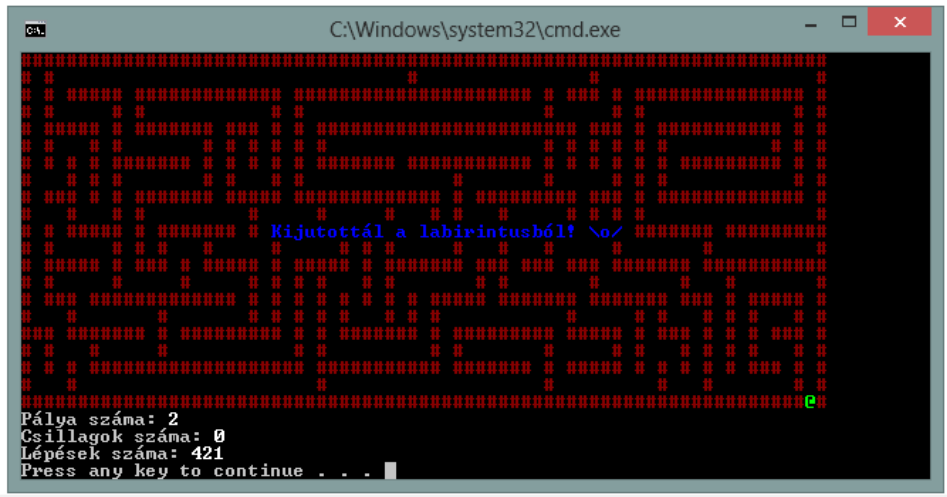
Ezt követően a játék elindul. A játékost a jobb felső sarokban megjelenő zöld '@' karakter szimbolizálja. A csillagok számának 0-ra csökkenésekor kimehetünk az ajtón (jobb alsó sarokban található '=' karakter) (13., 14. és 15. ábra)



13. ábra



14. ábra



15. ábra

4. Felhasznált irodalom:

Design your own maze:
http://gwydir.demon.co.uk/jo/maze/design/index.htm

+1 Egyéb:

White Rabbit font:
http://www.dafont.com/white-rabbit.font