**Keverintus Programozás 3 házi feladat**

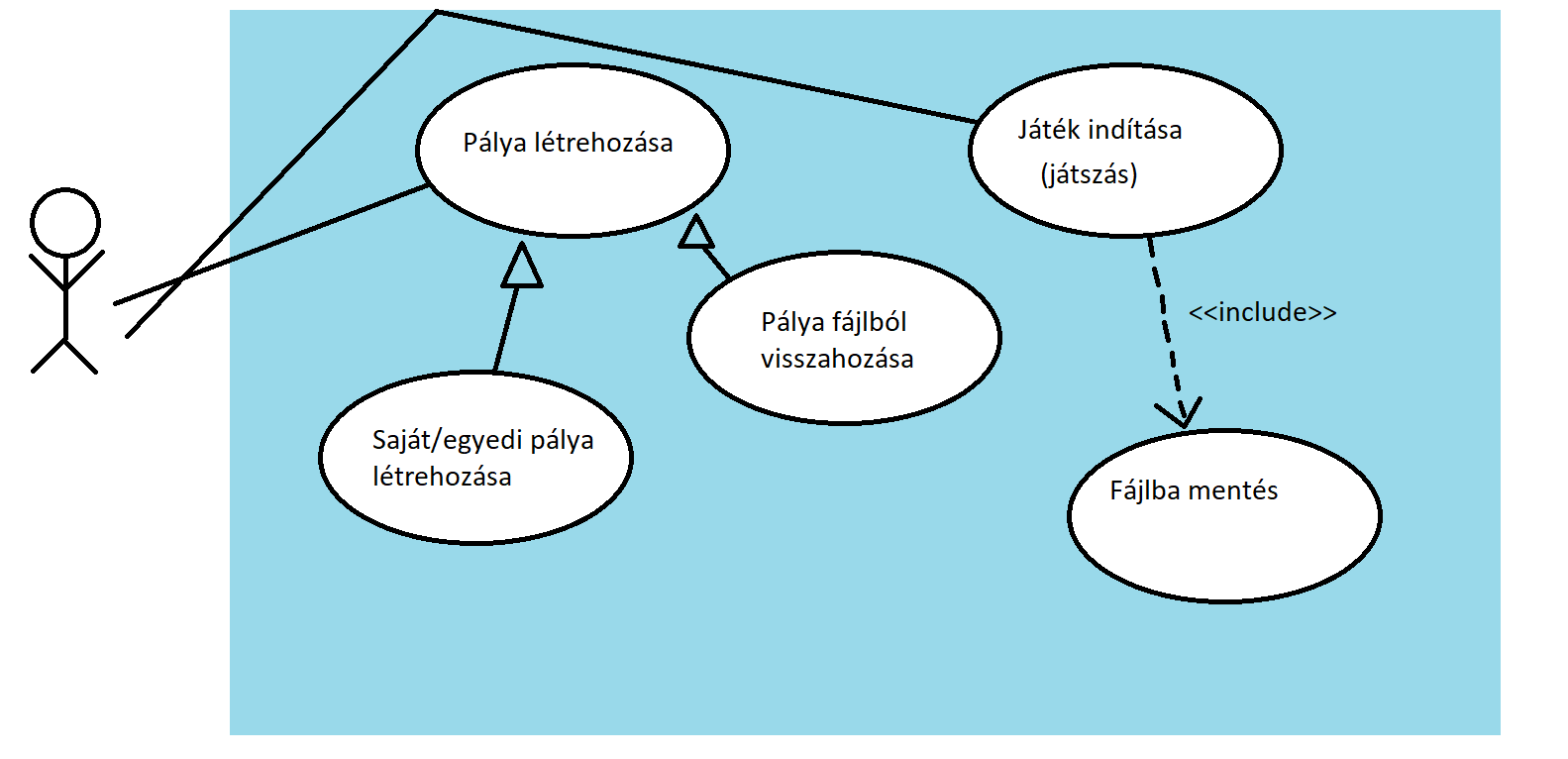
A játék elején a felhasználó, a megjelenő menüben beállíthatja, hogy milyen paraméterekkel rendelkező pályát szeretne. Megadhatja, hogy hány szörny, hány kincs legyen a labirintusban és azt is, hogy mekkora legyen a pálya. (hány mezőből álljon a négyzet alakú pálya oldala) kiválaszthatja azt is a felhasználó, hogy egy régebbi játékot tölt be egy fájlból, ekkor megadhatja a fájl nevét. („regiJatek”) Ezen kívül ha saját maga szeretne pályát rajzolni, akkor a megadott pálya méretében egy fallal teli pályát kap amint a WASD gombokkal közlekedhet és amelyik mezőn áll, az kincsé, szörnyé üres mezővé, vagy újra fallá teheti.

Kincs: b  
Üres mező: v  
Fal: n  
Szörny: m

Ha készen van a pályával, akkor a K gombot kell lenyomni, hogy elkezdődhessen a játék.  
A játék során a kincseket össze kell szedni, a szörnyekkel nem szabad találkozni és ki kell jutni a labirintusból. A bábut a WASD gombokkal lehet irányítani. A szörnyek mozognak, szóval óvatosnak kell lenni. Ha játékállást el szeretné menteni a felhasználó, akkor az F gomb lenyomásával megteheti, és legközelebb a „regiJatek”-ot begépelve a megfelelő menüponthoz ugyanonnan lehet majd folytatni.

A játék közben a pálya fölött lesz egy státuszpanel, ami az aktuális koordinátait mondja meg a bábunak és azt, hogy hány kincset kell még összegyűjtenie. ha a koordináták -1, -1-re állnak, akkor vége a játéknak.

**Use-case-ek:**



**Pálya létrehozása:**  
A felhasználó egy random generált labirintust kap a megadott paraméterekkel.

**Saját/egyedi pálya létrehozása:**  
A felhasználó meg tudja rajzolni a saját pályáját a megfelelő paraméterekkel.

**Pálya fájlból visszahozása:**  
A felhasználó, a menüben a "regiJatek” szót begépelve vissza tudja hozni az előzőekben elmentett pályáját.

**Játék indítása:**  
A felhasználó elindíthat egy játékot, amit végigjátszik. Ebbe beletartozik a fájlba mentés is.

**Játék fájlba mentése:**  
A felhasználó a játék közben bármikor elmentheti az aktuális játékállást az F gomb lenyomásával.

**Class diagram:**

**Palya:**

kaputAllit()  
Random kiválaszt egy mezőt a pálya szélén, amit kapunak állít és onnan a megfelelő irányba lép egyet, ahonnan a virtuális bányász elkezdheti a labirintus készítését.

vajhate(int x, int y, int c)  
Megkap egy x és y koordinátát és egy irányt. A függvény megnézi, hogy az így meghatározott mezőt kivájhatja-e üres mezővé a virtuális bányász. Ha a labirintussűrűségnél töb szomszédja van akkor nem vájhat különben igen.

hanyUresSzomszedVan(int x, int y)  
Megadja, hogy a kapott kordinátákon található mezőnek, hány üres szomszédja van.

palyatIr()  
létrehoz egy random generált pályát a megfelelő paraméterek szerint.

palyatKiir()  
a szabványos kimenetre kiírja a pályát.

hanyUresMezoVan()  
megszámolja, hogy hány üres mező van a pályán és ezzel a számmal vissza is tér.

babubutElhelyez()  
Egy random választott üres mezőre elhelyezi a bábut és a babuX és babuY paramétereket beállítja a megfelelő koordinátákra.

leptetSegit(int x, int y)

a leptet() függvény használja és az ő feladata lett ketté bontva.

leptet(Irany irany)  
A bábut a megadott irányba lépteti, ha tud arra lépni.

szornyeketLerak()

A megadott számban random elhelyezi a szörnyeket üres mezőkre és a szornyek tömbben beállítja egyesével a koordinátáikat, hogy léptetésnél könnyű legyen megtalálni őket.

kincseketLerak()

A megadott számban lerakja random üresmezőkre a kincseket.

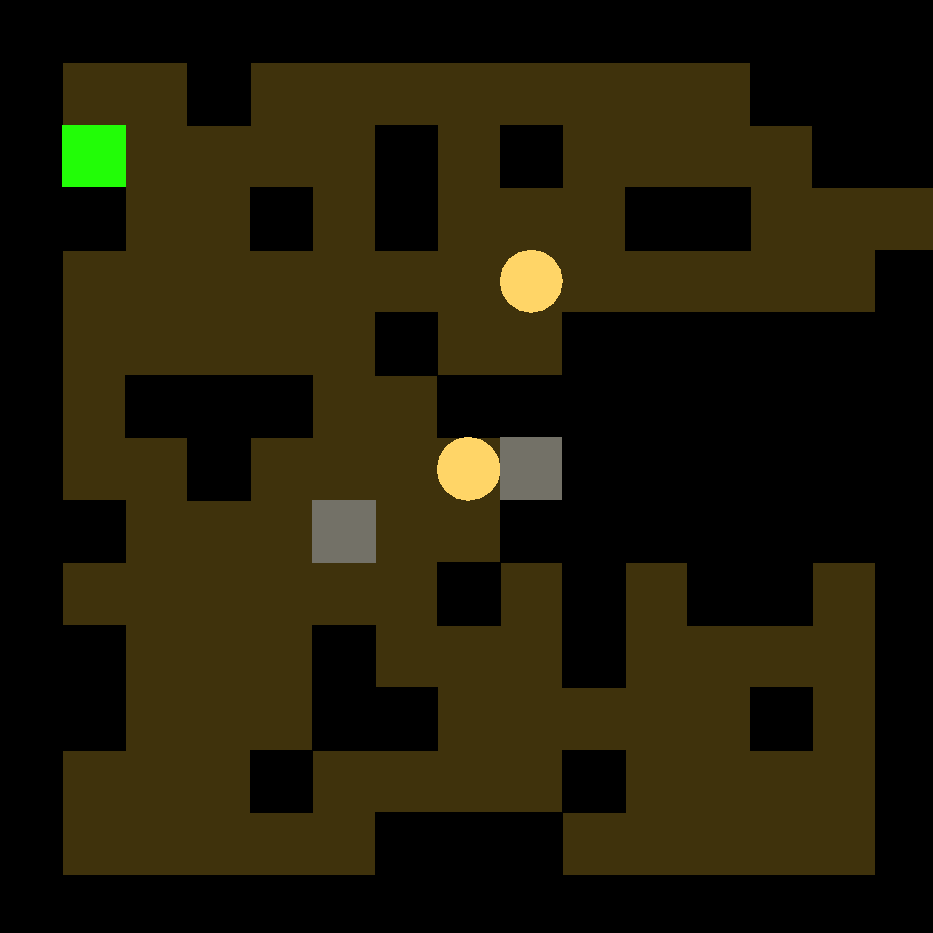
szornyeketLeptet()  
A szörnyeket egyesével egy véletlenszerűen generált irányba lépteti.

kincsekSzama()  
Megadja, hogy hány kincs van fenn a pályán.

setPalyaErtek(int x, int y, Mezo m)  
a megadott koodrináták helyén beállítja a mező típusát a megadott m típusra.

szornyeketBeallit()  
A pályán lévő szörnyeket belerakja a szornyek tömbbe, ahol a koordinátáik tárolódnak. (léptetésükhöz szükséges)

**JatekPanel**

draw()  
Kirajzolja a JatekPanelnek megadott pályát:  
****

**KezdoPanel()**

szazalekBeallitas()  
Beállítja a pálya méretétől függően azt a számot, amely megadja a palya írásakor, hogy hány százalékban legyen üres mező a pályán.

PalyatCsinal

kapuKeszE()  
Megvizsgálja, hogy van-e már kapu a pályán. Ha egynél több van, akkor az egyiket visszaállítja fallá, így a felhasználó mindig csak egy kaput tud beállítani a pályájának.

beallit2()  
A beallit metódusnak segít.

beallit()  
A kapott koordinátákon beállítja azt a mezőt, amit a felhasználó kiválasztott: kincs, fal szörny üres mező.