Dokumentáció

Készítették (ábécé sorrendben):

Ancsin Attila, Babati Bence, Dananaj Pál, Lengyel Mihály

# Követelmény feltárás

## Célkitűzés

A projekt célja egy ún. **scrolling shooter** játék készítése, mely az űrben játszódik. A játék célja az ellenséges űrhajók lelövése, saját űrhajónk épségben tartása az egyre növekvő nehézségi szintű pályákon.

### Játékmenet

A játékban egy űrhajót irányítunk, égérrel vagy billentyűzettel, mellyel tudunk mozogni és lőni. A képernyő folyamatosan halad előre a pálya felett, egészen a pálya végéig. A játékosnak csak a képernyőn belül van mozgástere, így kötelező az előrehaladás. Egy-egy pályát akkor teljesítettünk, ha eljutottunk a pálya végéig, és ha a pálya végén lévő esetleges **boss**-t megsemmisítettük.

A pályákon lehetnek különböző nem megsemmisíthető tereptárgyak, az ezekkel való ütközést kerülni kell. Játék közben az adott pályától függően jelennek meg az ellenséges űrhajók és egyéb tárgyak, melyek megsemmisítése esetén a játékos jutalmat, azaz pénzt és tapasztalati pontot kap, amiket később a fejlesztő képernyőn elkölthet. A különböző tárgyak és űrhajók megsemmisítése nagyrészt opcionális, de rengeteg előnyhöz juttathatja a játékost. Egyes ellenségek elpusztítása után felvehető tárgyak maradhatnak a pályán, melyek szintén a játékos előnyére válhatnak.

A játékos és az ellenséges űrhajók is lőhetnek játék közben, több fajta fegyvert használva. A játékos hajója rendelkezik bizonyos pajzzsal és páncéllal, ezek értéke a hajót eltaláló lövedékek hatására csökken. A lövések először a pajzs majd a páncél értékét csökkentik, de az előbbi kis idő múltán magától újratöltődik. Ha a hajó páncélja elfogyott, az megsemmisül és a játékos elbukja az adott pályát. Az egyes pályák teljesítése vagy elbukása után a játékos a fejlesztő képernyőre kerül, ahol az eredménytől függően a következő pályára léphet vagy újrajátszhatja az aktuálisat. A játék bármikor megállítható és újraindítható.

### Főmenü

A program indításakor a felhasználó egy menübe kerül, itt több lehetőség közül választhat: Új játék indítása, Játék folytatása, Beállítások és Toplista. A Beállítások és a Toplista menüpont a megfelelő képernyőre továbbítja a játékost, aki előbbin módosíthatja a játék különböző beállításait, az utóbbin pedig megtekintheti a toplistát, azaz a legsikeresebb játékosok listáját.

Új játék indítása esetén a játékos megadja a nevét és hajótípust választ, majd a játék indul az első pályával, míg a Játék folytatása esetén kiválaszthat egy korábbi mentést, ekkor az adott mentés következő pályája előtti fejlesztő képernyőre kerül a játékos. A különböző hajótípusok erejüket tekintve összességében azonosak, de különböző játékstílusokat tesznek lehetővé a játékos számára.

### Fejlesztő képernyő

A fejlesztő képernyő két pálya között a pilóta vagy az űrhajó fejlesztését teszi lehetővé. Az előbbire az előző pálya vagy pályák során szerzett tapasztalati pontokat, utóbbira pénzt használhatunk. Lehetőség van menteni az aktuális játékot, illetve újrajátszhatjuk az aktuális pályát, illetve ha azt már korábban teljesítettük, akkor továbbléphetünk a következőre.

A pilóta fejlesztését tulajdonság pontok vásárlásával tehetjük, ezek adott számú tapasztalati pontba kerülnek. Az egyes tulajdonságok a hajó több tulajdonságára is hatással lehetnek. Hatással lehetnek más dolgokra is, például az esetleges bonuszok esélyére, a szerzett tapasztalati pontok illetve pénz mértékére is.

A hajó fejlesztését különböző modulok vásárlásával és a hajóra csatolásával érhetjük el. A hajóra csatolható modulok száma és helye az adott hajótól függ. Egyes moduloknak lehetnek saját beállításai, mint például fegyver esetén beállítható a lövés szöge, illetve hogy lő-e automatikusan bekapcsolt folyamatos lövés esetén. Egy-egy modul csak akkor van hatással, ha a hajóra van csatolva. Az aktuálisan nem használt modulok sem vesznek el.

## Fogalomjegyzék

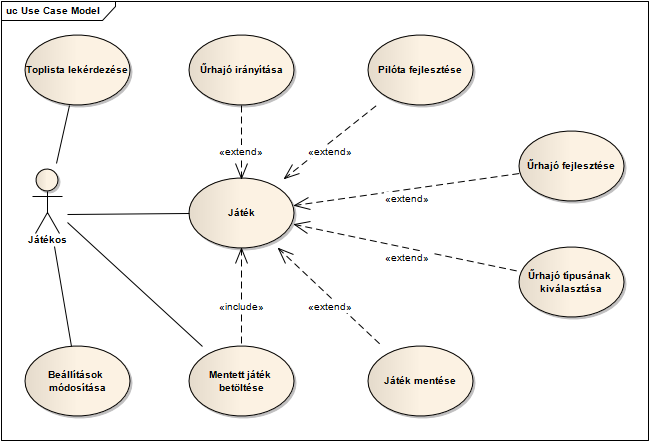
**Scrolling shooter:** gyors ütemű lövöldözős játék, sok ellenséggel. Fő ismérve, hogy a pálya folyamatosan előrehalad a játékos hátterében, ezáltal a játékos előrehaladása is kötelező.

**Boss**: egy pálya végén található nagy, erős ellenséges egység, amit el kell pusztítania a játékosnak a pálya befejezéséhez.

**Mob**: moving object - az elpusztítható tárgyakat és hajókat jelöli.

## Funkcionális követelmények

A játék funkcionális követelményeit az alábbi használati eset diagram mutatja.



1. ábra: Használati eset diagram

**A használati eset neve:** Toplista lekérdezése

**Felelős:** XY

**Leírás:** Az eddigi játékosok legjobb eredményeit mutató Toplista megjelenítése a felhasználó számára. A Toplista adatokat adatbázisban rögzítjük. Csak a legjobb 10 eredményt tároljuk.

**Előfeltétel:** Adatbázis megléte, álló játék.

**Utófeltétel:** A képernyőn megjelenik a toplista a legjobb 10 játékos nevével, valamint eredményével az adatbázisban tárolt aktuális adatoknak megfelelően.

**A használati eset neve:** Beállítások módosítása

**Felelős:** XY

**Leírás:** A játékos módosíthatja a játék, vagy a képernyő bizonyos előre meghatározott jellemzőit. Az aktuális beállításokat egy adatbázisban tároljuk.

**Előfeltétel:** Adatbázis megléte.

**Utófeltétel:** Adatbázisban tárolt adatok aktualizálódnak a felhasználó által beállított értékeknek megfelelően.

**A használati eset neve:** Játék

**Felelős:** XY

**Leírás:** A fő használati eset. A játék a *Játékmenet* c. fejezetben leírtak szerint zajlik. A játékmenetre vonatkozó beállításokat adatbázisból olvassuk be.

**Előfeltétel:** Adatbázis megléte.

**Utófeltétel:** A játék a játékmenetben megfogalmazottak szerint lezajlott.

**A használati eset neve:** Játék mentése

**Felelős:** XY

**Leírás:** A jelenlegi játék állását elmentjük egy adatbázisba. Így az később folytatható. Csak a pályák előtt vagy után van lehetőség mentésre, közben nincsen.

**Előfeltétel:** Elindult játék. Két pálya közötti állapotban van a játék.

**Utófeltétel:** A játék tulajdonságai (űrhajó tulajdonságai, stb.) rögzítve lettek az adatbázisban.

**A használati eset neve:** Űrhajó irányítása

**Felelős:** XY

**Leírás:** A játék lényegi része, amikor az űrhajót irányítjuk. Lehetőség van jobbra, balra, előre illetve hátra felé mozgásra. A felhasználó különböző tűzparancsokat is kiadhat. Az irányítás billentyűzettel illetve egérrel is lehetséges.

**Előfeltétel:** Elindított pálya.(Nem szüneteltetett játék).

**Utófeltétel:** Az űrhajó a kiadott parancsnak megfelelően reagál.

**A használati eset neve:** Mentett játék betöltése

**Felelős:** XY

**Leírás:** Lehetőség van korábban mentett játék betöltésére.

**Előfeltétel:** Álló játék.

**Utófeltétel:** A mentett játék betöltődik, a mentésnek megfelelő játék folytatható a következő pályával.

**A használati eset neve:** Pilóta fejlesztése

**Felelős:** XY

**Leírás:** A pályákon szerzett pontjainkat a pilótánk képességeinek növelésére költhetjük.

**Előfeltétel:** Álló játék.

**Utófeltétel:** A megvásárolt fejlesztéseknek megfelelően módosulnak a pilóta tulajdonságai. A következő pályán ezeknek megfelelően irányítható az űrhajó.

**A használati eset neve:** Űrhajó fejlesztése

**Felelős:** XY

**Leírás:** A pályákon szerzett pontjainkat az űrhajónk tulajdonságainak növelésére költhetjük (fegyverek, páncélzat, stb.)

**Előfeltétel:** Álló játék.

**Utófeltétel:** A megvásárolt fejlesztéseknek megfelelően módosulnak az űrhajó tulajdonságai. A következő pályán ezeknek megfelelően irányítható az űrhajó.

**A használati eset neve:** Űrhajó kiválasztása

**Felelős:** XY

**Leírás:** Új játék kezdése után ez a legelső lépés. Kiválasztjuk az űrhajót, amivel meg szeretnénk kezdeni a játékot.

**Előfeltétel:** Új játék kezdése.

**Utófeltétel:** A kiválasztott űrhajóval megkezdhető a tényleges játék.

## Nem funkcionális követelmények

**Fejlesztési módszertan:** Vízesés modell

**A fejlesztéshez szükséges hardver:**

**A fejlesztéshez használt szoftverek:**

* *Operációs rendszer:* Windows XP, 7, 8
* *Követelmény elemzés:*Word szövegszerkesztővel, LibreOffice Writer
* *CASE eszköz:* Enterprise Architect
* *C# fejlesztőeszköz:* Visual Studio 2012

**A futtatáshoz szükséges operációs rendszer:** Windows XP, 7, 8

**A futtatáshoz szükséges hardver:** Operációs rendszerek szerint megadva

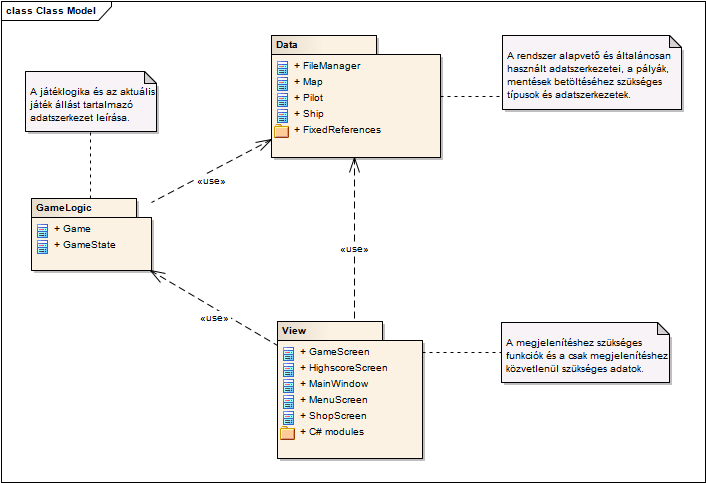
**Egyéb követelmények:** Intuitív felhasználói felület, könnyű kezelhetőség

# Tervezés

## Osztálymodell

### Felépítés

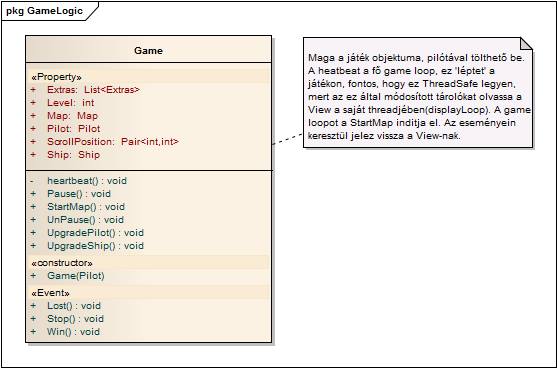
Az osztályok, csomagok alapvető szerkezetét a következő diagram mutatja be.



2. ábra: Összefoglaló osztálydiagram

### Játéklogika

A játéklogikát megvalósító GameLogic csomag felépítése az alábbi osztálydiagram mutatja be.

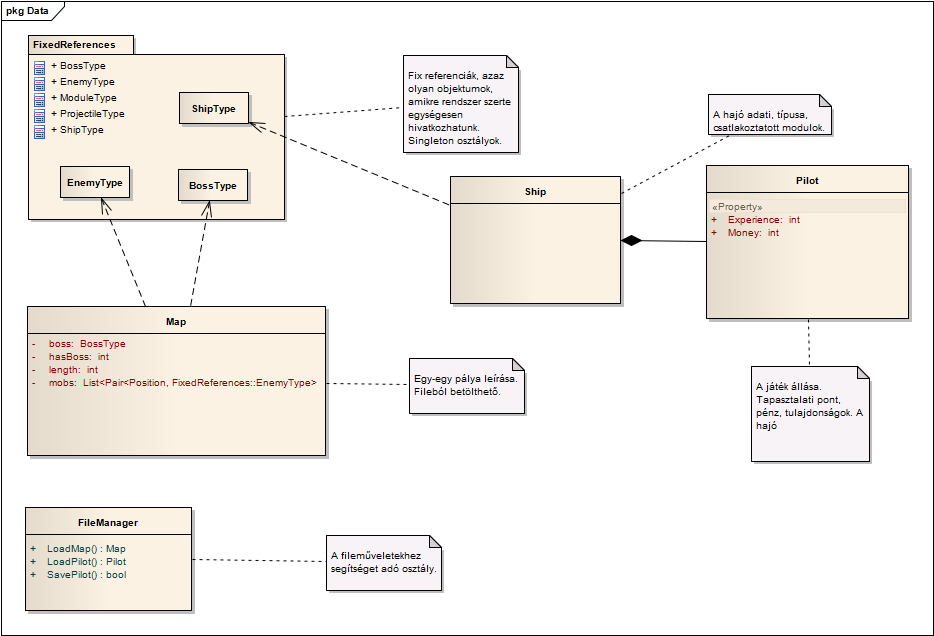


3. ábra: GameLogic osztálydiagram

Maga a játék objektuma, pilótával tölthető be. A heartbeat a fő game loop, ez 'léptet' a játékon, fontos, hogy ez ThreadSafe legyen, mert az ez által módosított tárolókat olvassa a View a saját threadjében (displayLoop). A game loopot a StartMap indítja el. Az osztály az eseményein (Win, Lost, Stop) keresztül jelez vissza a View-nak. Lehetőség van a játék szüneteltetésére illetve folytatására, ezeket a Pause illetve UnPause metódusok végzik. A pálya végén a játékos fejlesztheti a hajóját illetve pilótáját (UpgradePilot(), UpgradeShip() eljárások). A játék működéséhez szükséges paraméterek: a térkép (Map), a szint száma (Level), a pilóta (Pilot), a kiválasztott hajó (Ship). Továbbá az ellenfelek által „dobott” felvehető extrák (Extras).

### Adatkezelés

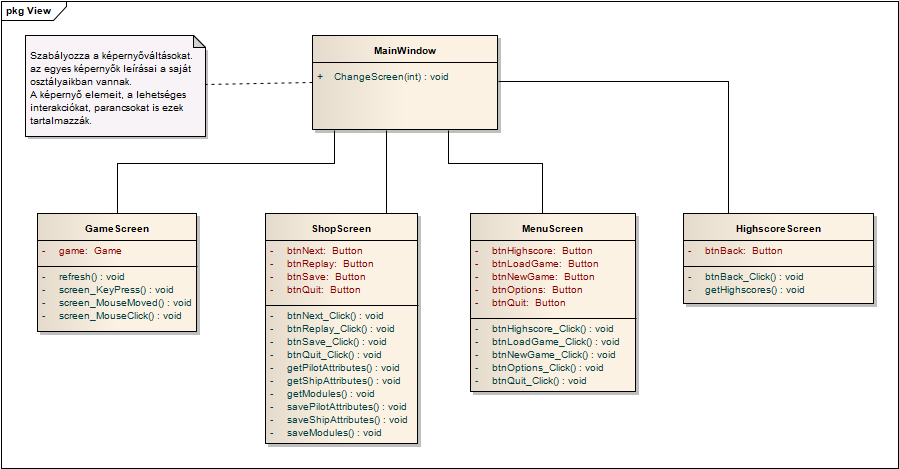
Az adatkezelő Data csomag szerkezetét a következő diagram mutatja be.



4. ábra: Data csomag osztálydiagramja

### Nézet

A játék nézetét képező osztályok szerkezete és kapcsolatai a következő diagramon találhatóak.



5. ábra: View csomag osztálydiagramja

A **View** csomag öt osztályból áll: az alkalmazás fő képernyőjéből, melyet a **MainWindow** osztály valósít meg, a játék menüjét kezelő **MenuScreen** osztályból, a toplistát mutató **HighscoreScreen** osztályból, a fejlesztésért felelős **ShopScreen** osztályból és a játékot megjelenítő **GameScreen** osztályból.

#### MainWindow osztály

Az alkalmazás fő képernyője, ez az osztály felel a megfelelő képernyők megjelenítéséért és a képernyőváltásokért.

##### Metódusok

**Név:** ChangeScreen(int) : void

**Leírás:** A képernyő váltásért felelő metódus, bezárja az aktuális képernyőt és a paraméterként kapott egész kód alapján létrehozza a megfelelő képernyő példányát.

#### MenuScreen osztály

Ez a képernyő tartalmazza a játék főmenüjét. A játékos választhat új játék indítása, a toplista mutatása, a beállítások módosítása, mentett játék betöltése és az alkalmazás bezárása opciók közül.

##### Adattagok

**Név:** btnNewGame: Button

**Leírás:** Az új játék indításáért felelős gomb.

**Név:** btnHighscores: Button

**Leírás:** A toplista mutatásáért felelős gomb.

**Név:** btnLoadGame: Button

**Leírás:** Mentett játék betöltéséért fellős gomb.

**Név:** btnOptions: Button

**Leírás:** A beállítások módosításáért felelős gomb.

**Név:** btnQuit: Button

**Leírás:** Bezárja az alkalmazást.

##### Metódusok

**Név:** btnNewGame\_Click(): void

**Leírás:** A btnNewGame gomb eseménykezelője. Bezárja a menüt és megjeleníti a fejlesztőképernyőt az alapbeállításokkal.

**Név:** btnLoadGame\_Click(): void

**Leírás:** A btnLoadGame gomb eseménykezelője. Bezárja a menüt és megjeleníti a fejlesztőképernyőt, melyen a mentett játék szerint folytathatjuk a fejlesztést.

**Név:** btnHighscores\_Click(): void

**Leírás:** A btnHighscores gomb eseménykezelője. Bezárja a menüt és megjeleníti a toplistát.

**Név:** btnQuit\_Click(): void

**Leírás:** A btnQuit gomb eseménykezelője. Bezárja az alkalmazást.

#### HighscoreScreen osztály

Ez az osztály felelős a toplista megmutatásáért.

##### Adattagok

**Név:** btnBack: Button

**Leírás:** A menübe való visszalépést biztosító gomb.

##### Metódusok

**Név:** btnBack\_Click(): void

**Leírás:** A btnBack gomb eseménykezelője. Bezárja a toplista képernyőjét és újra megjeleníti a menüképernyőt.

**Név:** getHighscores(): void

**Leírás:** Lekérdezi a toplista adatokat az adatbázisból.

#### GameScreen osztály

Ez az osztály felel a játék megjelenítésért és irányítást biztosít a játékosnak az úrhajó felett (egér és billentyűzet).

##### Adattagok

**Név:** game: Game

**Leírás:** A játékot megvalósító osztály példánya.

##### Metódusok

**Név:** refresh(): void

**Leírás:** A játékképernyő frissítését végzi a mögötte futó játéklogikának megfelelően.

**Név:** screen\_KeyPress(): void

**Leírás:** A billentyűzet eseménykezelője. A különböző billentyűk lenyomására különbözően reagál.

**Név:** screen\_MouseClick(): void

**Leírás:** Az egérgombok eseménykezelője. A különböző egérgombok lenyomására különbözően reagál.

**Név:** screen\_MouseMoved(): void

**Leírás:** A egérmozgatás eseménykezelője.

#### ShopScreen osztály

Ez az osztály felel a fejlesztő képernyő megjelenítésért. Megjeleníti az űrhajó, a pilóta és a modulok jelenlegi állapotát és elmenti a játékos változtatásait. A képernyőről folytathatjuk, menthetjük és bezárhatjuk az aktuális játékot, vagy újrajátszhatjuk az előző pályákat.

##### Adattagok

Név: btnNext: Button

**Leírás:** A következő pályára indításáért felelős gomb.

**Név:** btnSave: Button

**Leírás:** A játék mentéséért felelős gomb.

**Név:** btnReplay: Button

**Leírás:** A pálya újrajátszását biztosító gomb.

**Név:** btnQuit: Button

**Leírás:** A játék bezárásáért felelős gomb.

##### Metódusok

**Név:** btnNext\_Click(): void

**Leírás:** A btnNext gomb eseményekezelője. Bezárja a fejlesztőképernyőt és megjeleníti a játékképernyőt a következő pályával.

**Név:** btnSave\_Click(): void

**Leírás:** A btnSave gomb eseménykezelője. Elmenti az aktuális játékot.

**Név:** btnReplay\_Click(): void

**Leírás:** A btnReplay gomb eseménykezelője. Bezárja a fejlesztőképernyőt és megjeleníti a játékképernyőt az előző pályával.

**Név:** btnQuit\_Click(): void

**Leírás:** A btnQuit gomb eseménykezelője. Bezárja a fejlesztőképernyőt és megjeleníti a menüképernyőt.

**Név:** getPilotAttributes(): void

**Leírás:** Lekérdezi a pilóta állapotát az adatbázisból és megjeleníti a képernyőn.

**Név:** getShipAttributes(): void

**Leírás:** Lekérdezi a hajó állapotát az adatbázisból és megjeleníti a képernyőn.

**Név:** getModuleAttributes(): void

**Leírás:** Lekérdezi a jelenlegi modulokat és állapotukat (használatban van-e) és megjeleníti a képernyőn.

**Név:** savePilotAttributes(): void

**Leírás:** Elmenti a pilóta állapotát az adatbázisba.

**Név:** saveShipAttributes(): void

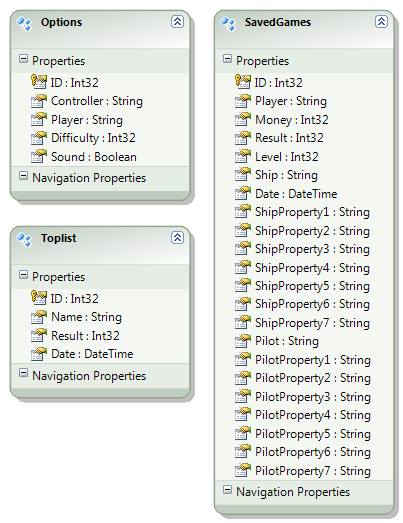
**Leírás:** Elmenti a hajó állapotát az adatbázisba.

**Név:** saveModuleAttributes(): void

**Leírás:** Elmenti a modulok állapotát az adatbázisba.

### Adatbázisterv

Az adatbázis modelljét a következő diagram mutatja be.



6. ábra: Adatbázisterv

táblában tároljuk a játék működéséhez szükséges adatokat. Az Options tábla az alapbeállításokat tárolja (Nehézség, Hang, Irányítás, Játékos neve). Ezek alapértelmezett értékekkel rendelkeznek, és egyik mező sem lehet üres. A tábla csupán egyetlen sort tartalmaz.

A Toplist tábla tárolja az eddigi legjobb eredményt elérő játékosok nevét és az elért pontszámukat, valamint hogy mikor játszottak. A tábla maximum 10 sort tartalmazhat.

A SavedGames tábla a mentett játékokat tárolja. Szükséges tárolni a Játékos nevét, eddig elért pontszámát, melyik pilótát, illetve hajót választotta, továbbá azok milyen tulajdonságokkal rendelkeznek. Fontos továbbá, hogy hányas szinten tart a játékos, illetve mennyi pénzzel rendelkezik. Rögzítjük továbbá a játék mentésének idejét.

### Felhasználói felület

Kezdeti menü:  


Indítás után, játék közben:  


A pályák közötti fejlesztő ablak:  


A legjobb eredmények megtekintéséhez felület: