Hálózati ulti program

Bagó Tamás
(toto@elte.hu)
programtervező matematikus, nappali tagozat

Témavezető: Istenes Zoltán

Eötvös Loránd Tudományegyetem, Informatikai Kar
Budapest, 2004

Tartalomjegyzék

1. Felhasználói dokumentáció

1.1. Bevezető

A program a Magyarországon nagy népszerűségnek örvendő ulti nevű kártyajátékot valósítja meg hálózati környezetben. Ennek segítségével egymástól távol lévő baráti társaságok is hódolhatnak a játéknak az interneten keresztül. Használatához egy nem túl erős, hálózatra kötött számítógépre, és kettő másik hasonlóan felszerelt játékosra van szükség.

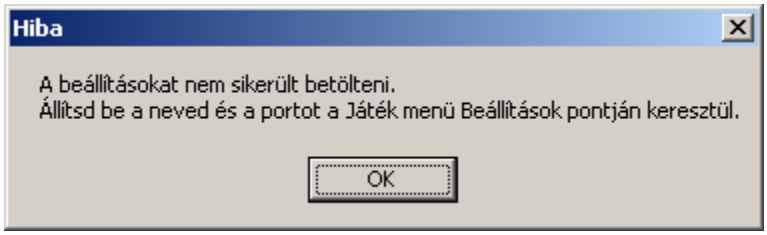
A program ezen kívül kezdőknek segíthet a játék szabályainak elsajátításában, mivel nem engedi meg nekik, hogy szabálytalan lépést hajtsanak végre. Így ha a gép nem reagál a felhasználó utasítására, akkor ő tudhatja, hogy ez a lépés nem megfelelő. Licit során is megtanulhatja az egyes játékok értékeit, hiszen ezek mindig kijelzésre kerülnek, illetve azt is, hogy melyik játékot melyikre lehet rámondani (néha azonos értékűek közt is lehet különbség), ugyanis addig nem engedi tovább a licitálást, amíg nagyobbat nem mond. A kártyákat is a játék „színességének” (színes vagy színtelen) megfelelően rakja sorba a lejátszás során, ezzel is segítve a felhasználót.

A játékot az ulti – általam ismert – szabályai alapján játszák, amely néhány ponton eltérhet a máshol létező szokásoktól (lásd melléklet).

A szoftver grafikus (ablakos) felülettel rendelkezik, kezelése egyszerűen egérrel történik. Törekedtem a könnyű használhatóságra és az átláthatóságra.

1.2. A program telepítése, rendszerkövetelmények

A programot (ulti.exe) egyszerűen a CD-ROM-ról másoljuk a merevlemezre, egy általunk kiválasztott tetszőleges könyvtárba. Az első futtatáskor hibaüzenetet fog adni, hogy nem sikerült betöltenie a beállításokat (ugyanis még nem létezik a megfelelő file), és felhívja a figyelmünket, hogy mindent állítsunk be. (Erről részletesebben később.)

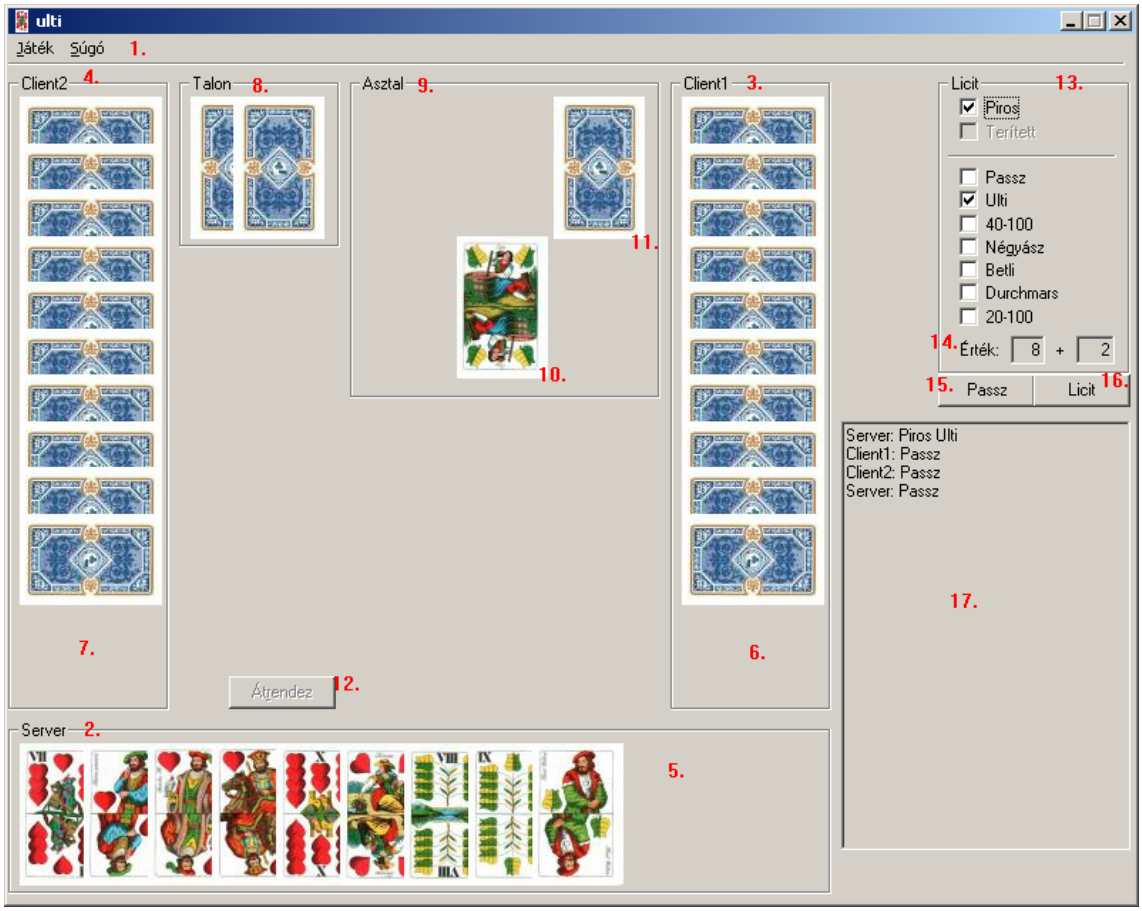


A program futtatása 32 bites Windows operációs rendszerben lehetséges.

A játék használatához legalább Pentium 200 MHz-es processzor, 32 MB memória ajánlott. Ezen kívül szükséges, hogy a gép rendelkezzen valamilyen hálózati kártyával és egérrel az alapvető eszközökön kívül.

A teljes áttekinthetőség érdekében ajánlott legalább 1024x768-as felbontás használata.

1.3. A felület felépítése



A főablak elemei:

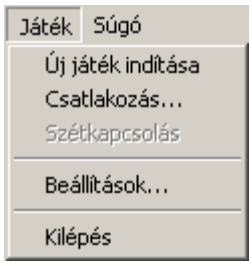
1. Menüsor.
2. Az aktuális felhasználó neve.
3. Az első ellenfél neve.
4. A második ellenfél neve.
5. Az aktuális felhasználó lapjai.
6. Az első ellenfél lapjai.
7. A második ellenfél lapjai.
8. Talon.
9. Asztal. Ide kerülnek játék közben a lerakott lapok.
10. Az aktuális játékos által utoljára lerakott lap.
11. Jelzi, hogy melyik játékos következik.
12. Átrendezés gomb. Ennek segítségével rakosgathatjuk a tízeseinket a király felettről az alsó alá és vissza.
13. Licit mező. Itt végezhetjük el a jelölőnégyzetek segítségével a licitálást.
14. A kiválasztott játék értéke.

15. Passz gomb. Ha nem akarunk „nagyobbat mondani”, akkor ezzel passzolhatunk, illetve először a Passz-Licit kombináció helyett is használható.
16. Licit gomb. Miután kiválasztottuk, mit szeretnénk játszani, ezzel tudathatjuk a többiekkel, miután leellenőrizte, hogy „elég nagyot mondtunk-e”.
17. Üzenet mező. Itt jelennek meg az általunk és az ellenfelek által generált üzenetek, amik lényegesek a felhasználók számára.

1.4. A menü

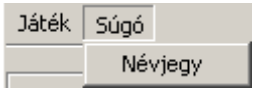
1.4.1. A játék menü

Innen érhetjük el a következő menüpontokat:

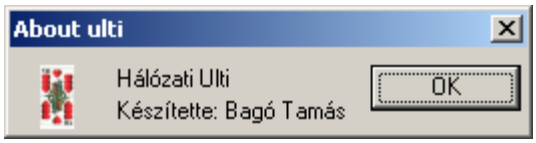


- *Új játék indítása:*
  Segítségével új játékot indíthatunk, ahol mi leszünk a szerver. Fontos ebben az esetben, hogy ismerjük gépünk IP címét vagy kanonikus nevét, és hogy ezt tudassuk a csatlakozni kívánó többi emberrel.
- *Csatlakozás:*
  Ezzel kapcsolódhatunk egy már elindított szerverhez. Lényeges, hogy ismerjük a szerver IP címét vagy kanonikus nevét, illetve hogy ugyanazt a portot állítsuk be kommunikációra, mint a szerver.
- *Szétkapcsolás:*
  Ezzel a ponttal kapcsolódhatunk szét a szervertől, illetve állíthatjuk le a szervert, ha mi indítottuk. Akkor válik aktívvá, ha fut a szerverünk, vagy már csatlakoztunk hozzá.
- *Beállítások:*
  Itt érhetjük el a beállítások dialógusablakot, ahol a nevünket és a csatlakozáshoz használt portot állíthatjuk be.
- *Kilépés:*
  Lezár minden aktív kapcsolatot, majd kilép a programból.

1.4.2. A súgó menü

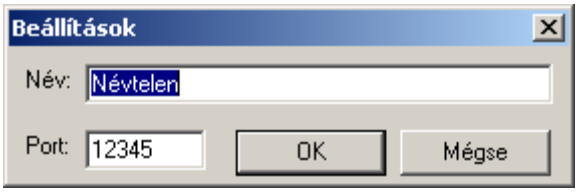


Itt találhatjuk a Névjegy menüpontot, amely megmutatja nekünk a program és a készítő nevét.



1.5. A játék előtt

Nagyon fontos, hogy mielőtt elkezdenénk játszani, helyesen kitöltsük a beállításokat. Ezt a Játék menü Beállítások pontjában tehetjük meg.

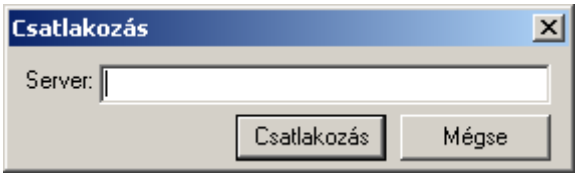


Ezeket a beállításokat elég egyszer megtennünk, mert ezeket a program eltárolja nekünk az *ulti.cfg* nevű állományban, és a következő indításkor automatikusan betölti. Ha a betöltés valamilyen okból mégsem sikerült volna (pl. a konfigurációs file nem található), akkor erről hibaüzenetet kapunk, és a program alapértelmezett értékekkel tölti ki a megfelelő mezőket.

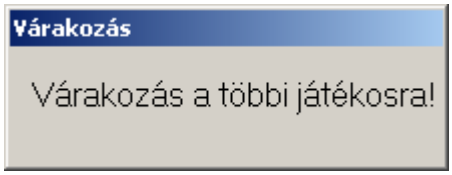
A sikeres csatlakozáshoz szükséges, hogy minden játékos ugyanazt a portot adja meg, és hogy ez olyan érték legyen, amit semmi más nem használ.

Lényeges, hogy minden játékos különböző nevet adjon meg, enélkül a program helyes működése nem biztosított.

A játék elindításához az egyik résztvevőnek el kell indítania a szervert. Ezt a Játék menü Új játék indítása pont segítségével teheti meg. A többieknek ehhez a szerverhez kell csatlakozniuk a Csatlakozás… menüponttal. Ezután be kell írni a megjelenő ablakba a szerver címét (a gyorsabb csatlakozáshoz érdemes IP címet megadni), majd a Csatlakozás gombra kattintani.



Amíg nincs meg a megfelelő játékosszám (3), addig egy erre figyelmeztető ablak jelenik meg.



1.6. A felhasználói felület kezelése

A játék nagy részben kizárólag egérrel is használható. Ez alól kivétel, amikor egy szöveges mezőt kell kitölteni (név, port, szerver címe). Ezen kívül a játék nagy részben billentyűzettel is kezelhető (kivétel a kártyák), úgy hogy lenyomjuk az ALT billentyűt, és vele egyidejűleg a kívánt mező nevében található aláhúzott betűt, számot.

A talon felvételéhez valamely, a talonban található lapra kell kattintani. Ekkor a lent lévő összes lapot felvesszük.

Egy kártya lerakásához rá kell kattintani egérrel. Ha ekkor nem történik semmi, akkor az azt jelenti, hogy nem a szabályoknak megfelelően léptünk. Ekkor egy új lappal kell próbálkozni. Érdemes a lap közepét „megcélozni”, ugyanis – főleg a bal – széle felé előfordulhat, hogy a szomszédos kártyát teszi le ahleyett, amit mi szerettünk volna. Ezt a határt licitálás alatt lehet veszélytelenül kipróbálni, ugyanis ekkor még visszaveheljük a lapjainkat.

1.7. A játék

Ha mindenki csatlakozása sikeres volt, a játék elindul. A szerver kever, és elküldi a lapokat a klienseknek. A licit elkezdődik (első körben mindig a szerver kezd).

A licit és a játék során is a játékosok óramutató járásával ellentétes irányú sorrendben követik egymást.

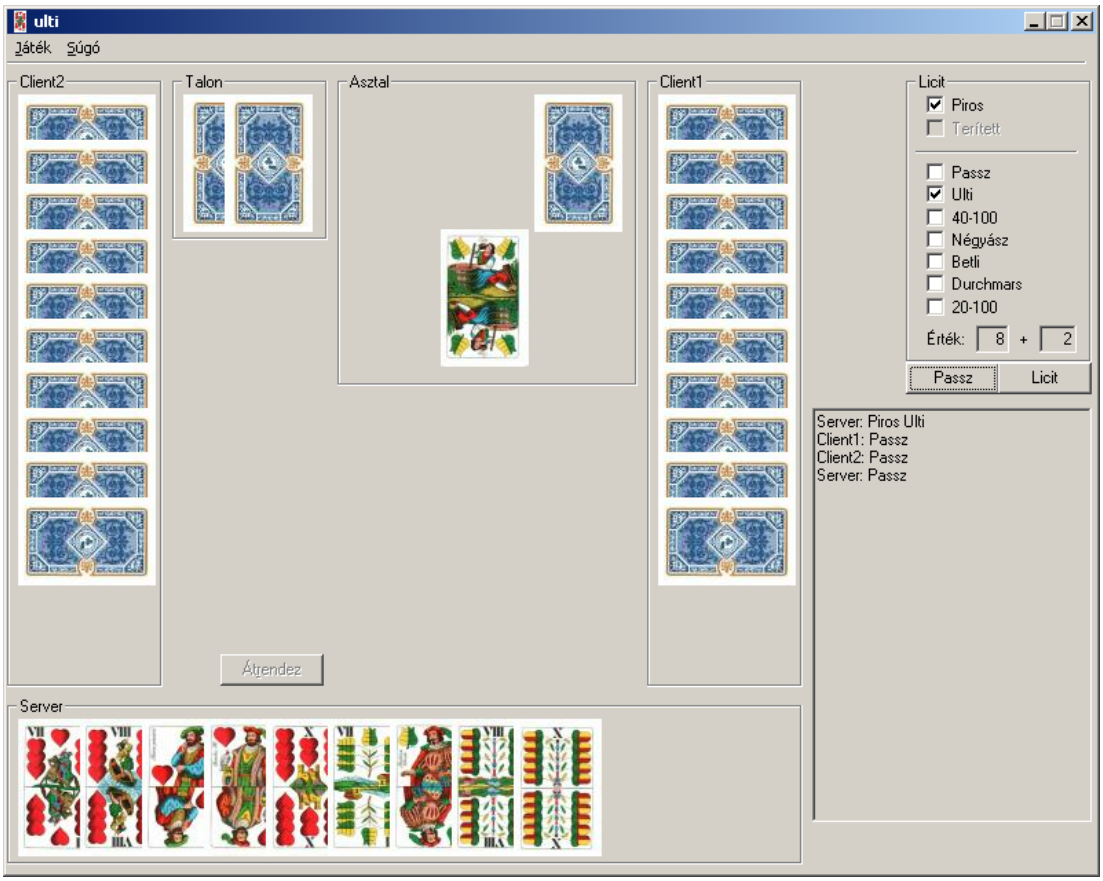
Licitálni csak akkor lehet, ha a soron következő játékos lerakott már két lapot a talonba, és a kiválasztott játék értéke nagyobb, mint az eddigi legnagyobb bemondás, vagy ugyanolyan nagyságú, de ez piros. Az ulti plussz értéke arra szolgál, hogy kiemelje az azonos értékő játékok közül, de kisebb mint az a játék, amleynek csak az értéke már nagyobb, mint az ulti értéke plusszal együtt. Az éppen kiválasztott licit értékéről az érték mező tájékoztatja a felhasználót. Rossz licitet nem lehet megadni (pl.: ulti betli, vagy terített négyász).

Amíg a licit folyik, a felhasználó átrendeztetheti a lapjait, annak megfelelően, hogy színes vagy színtelen játékot szeretne játszani. A tényleges játék kezdetén ez automatikusan beállítódik a játék fajtájától függően.

Ha „elhangzott” három passz (a kezdő játékos legelső passza játéknak számít), elkezdődik a legnagyobb licit lejátszása. A soron következő játékost az asztalon egy lefelé fordított kártyával jelzi – kivétel új kör elején, hogy látni lehessen a lapokat. A színre-szín és a kötelező felülütés szabályokat a gép betartatja, azaz addig nem enged rakni, amíg megfelelő lapot nem választunk.

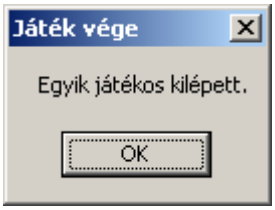
Húszakat vagy negyvenet nem kell bejelenteni, ezeket a gép a kör elején automatikusan eltárolja, és a végén – ha szükséges – a játék teljesítésének megállapításához felhasználja.

Színes játék kezdetén – ha nem piros – a program rákérdez az adu színére. Miután ez kiválasztásra került, mindenki értesül róla.



Ha a játék befejeződött, a program kiértékeli a végeredményt. Ezután új leosztás kezdődik, és az első licitáló a következő játékos lesz, az új keverés után ő kap több lapot, ő kezdheti el a bemondásokat.

A teljes játék abban az esetben fejeződik be, ha valamelyik játékos kilép, vagy a Szétkapcsolás menüpont segítségével megszakítja a kapcsolatot. Erre a többieket egy felugró ablak figyelmezteti.



2. Fejlesztői dokumentáció

A program Windows operáció rendszer alatt Microsoft Visual C++ 6.0-val készült. Használja a fejlesztőkörnyezet által nyújtott MFC osztályokat, amihez szükséges könyvtárat statikusan linkeli hozzá, így futtatásához semmi egyebet nem kell telepíteni.

Tetszőleges 32 bites Windows rendszeren futtatható.

2.1. A CAboutDlg osztály

Az osztály a Súgó menü Névjegy pontjára való kattintáskor megjelenő ablakot valósítja meg. Ez a program és készítőjének nevét tudatja a felhasználóval. A hozzá tartozó dialógus erőforrást én tervezten, az osztályt a fejlesztőkörnyezet automatikusan generálta.

2.2. A CAduDlg *osztály*

Színes játék esetén ez az osztály/ablak ad lehetőséget az adu színének kiválasztására. A hozzá tartozó dialógus erőforrást én tervezten, az osztályt a fejlesztőkörnyezet automatikusan generálta.

2.3. A CClient *osztály*

Itt valósul meg a kliens osztály. Ezen keresztül tudnak a kliensek a szerverrel kommunikálni, illetve a szerver is ilyen adattagokon keresztül áll kapcsolatban a kliensekkel.

Adattagjai:

- *CUltiDlg *m_Dlg:*
  Mutató a „szülő” ablakra, azaz a program főablakára.
- *CSocketFile *m_File:*
  A kommunikációs sockethez rendelt file. Konstruktorát a **this** előre definiált változóval kell hívni.
- *CArchive* m_ArchOut:*
  Ezen keresztül lehet írni a socket-file-ra. Konstruktorát az *m_File* változóval kell hívni.
- *CArchive* m_ArchIn:*
  Ezen keresztül lehet olvasni a socket-file-ról. Konstruktorát az *m_File* változóval kell hívni.

Függvényei:

- *CClient(CUltiDlg *dlg):*
  Konstruktor. Paramétere a „szülő” ablak konkrét példánya.
- *~CClient():*
  Destruktor.
- *void Init():*
  A kliens inicializálását végző függvény. A sikeres csatlakozás után hívandó. Beállítja az *m_File, m_ArchIn* és *m_ArchOut* változókat.
- *void OnReceive(int nErrorCode):*
  Akkor hívódik meg, ha a kliens üzenetet kap. Paramétere a hiba kódját tartalmazza (ha nincs hiba, akkor 0).
- *void ReceiveMsg(CMsg *msg):*
  A kapott üzenetet betölti a paraméterében megadott tárolóba.
- *void SendMsg(CMsg *msg):*
  Elküldi az *msg* üzenetet a már kiépített kapcsolaton keresztül.

2.4. A CConnectDlg *osztály*

Ez az osztály/ablak valósítja meg a szerver címének bekérését csatlakozáskor. A hozzá tartozó dialógus erőforrást én tervezten, az osztályt a fejlesztőkörnyezet automatikusan generálta.

2.5. A CMsg *osztály*

Ilyen típusú üzeneteket lehet küldeni, fogadni. A hálózati kommunikáció során csak ennek az osztálnak a példányai kerülnek át egyik gépről a másikra.

Adattagjai:

- *CString m_Name:*
  A küldő neve.
- *UINT m_Type:*
  A küldött üzenet típusa.
- *UINT m_Message:*
  A küldött üzenet tartalma.

Fontos: mindig minden adattagot ki kell tölteni.

Függvényei:

- *CMsg():*
  Konstruktor.

- *~CMsg():*
  Destrukor.
- *void Serialize(CArchive& ar):*
  A paraméter típusától függően ír bele vagy olvas *ar*-ből. (Itt történik a hálózati kommunikáció megvalósítása). Az *ar* a mi esetünkben vagy *m_ArchIn* vagy *m_ArchOut*.

2.6. A CServer *osztály*

A szerver megvalósítása. Ezzel az osztállyal lehet elindítani, és ez végzi a kapcsolatok foagdását is. A kommunikációhoz a szerver is a *CClient* osztály egy-egy példányát használja.

Adattagja:

- *CUltiDlg *m_Dlg:*
  A „szülő” ablakra mutató pointer.

Függvényei:

- *CServer(CUltiDlg *dlg):*
  Konstruktor. Paramétere a „szülő” ablak konkrét példánya.
- *~CServer():*
  Destruktor.
- *void OnAccept(int nErrorCode):*
  Kapcsolódás esetén hívódik meg. Paramétere a hiba kódja (ha nincs hiba, akkor 0).

2.7. A CSetupDlg *osztály*

A beállítások megváltoztatásához szükséges osztály/ablak. A hozzá tartozó dialógus erőforrást én tervezten, az osztályt a fejlesztőkörnyezet automatikusan generálta.

2.8. A CUltiApp *osztály*

Maga az alkalmazás ennek egy példánya. A fejlesztőkörnyezet hozza létre. A hozzá tartozó dialógus erőforrást én tervezten, az osztályt a fejlesztőkörnyezet automatikusan generálta.

2.9. A CUltiDlg *osztály*

A program legfontosabb függvényeit és a főablakot megvalósító osztály.

Adattagjai:

- *CButton m_Atrendez:*
  Az Átrendez feliratú gomb ezen a változón keresztül manipulálható (engedélyez, letilt).
- *CButton m_..._Nev:*
  A játékosok nevei az ablakban.
- *CStatic m_...:*
  Az kártyalapok képeinek fenntartott helyek. Ezek fogadják a kattintásokat is.
- *BOOL m_Licit_...:*
  A licitáláshoz használt jelölőnégyzetek értéke.
- *UINT m_Ertek:*
  A licit értéke. Ez kerül kijelzésre az ablakban.
- *UINT m_Plussz:*
  A licit értékéhez tartozó plussz (pl.: ulti = 4+1) értéke. Ez kerül kijelzésre az ablakban.
- *CString m_Uzenet:*
  Az üzeneteket megjelenítő mező tartalma. Ez kerül kijelzésre az ablakban.
- *UINT m_Utolso_Utes:*
  Az utolsó kör legmagasabb kártyája. Segítségével megállapítható, hogy az ultit megnyerte-e a játékos vagy nem.
- *UINT m_Utolso_Visz:*
  Az utolsó kört vivő játékos. Segítségével megállapítható, kinek jár az utolsó ütésért járó +10 pont.
- *UINT m_Utesek[3]:*
  A játékosok által elvitt lapok. A játék eredményének kiértékelésében van szerepe (ütések értéke, betli, durchmars).
- *int m_Huszak[3]:*
  A játékosok húszai és negyvene (ha van). A parti győztesének megállapítását segíti.
- *UINT m_KiJatsza:*
  Ki játsza az aktuális játékot. Az utolsó licitáló sorszámát tartalmazza. Szükséges annak megállapításához, hogy ki kezdi a következő kört.
- *CStatic * m_Asztal[3]:*
  Az asztalon lévő kártyákra mutató pointer. Az asztal egyszerűbb kezelhetőségét hivatott segíteni.
- *UINT m_Jatek_Szine:*
  A játszott játék színének kódja. A négy kártyaszínen kívül színtelen (0x0000) értéket is felvehet.
- *int m_Passzok:*
  Hány passz „hangzott el” egymás után zsinórban. Ez alapján dönthető el, hogy végetért-e a licit. Nagyobb licit esetén mindig nullázódik.

- *UINT m_Jatek:*
  Mit játszunk. Az eddigi legnagyobb licit kódja.
- *bool m_isConnected:*
  Áll-e a kapcsolat.
- *UINT m_Kovetkezo_Jatekos:*
  Melyik játékos rak legközelebb.
- *bool m_Szintelen:*
  Színtelen-e a játék.
- *UINT m_Hanyadik_Kor:*
  Hányadik körben vagyunk (hány lapot raktak le eddig). A leosztás végének megállapítását segíti.
- *bool m_isLicit:*
  A licitálás fázisában vagyunk-e, vagy már játszunk?
- *UINT m_Jatek_Ertek:*
  Mennyi az eddig bemondott legnagyobb játék értéke.
- *UINT m_Kartyak_Asztal[3]:*
  Az asztalon lévő kártyák értékeit (típusát) tartalmazó tömb.
- *CBitmap m_CardBitmaps[66]:*
  A kártyák kirajzolásához használt bitmap-eket tartalmazó tömb.
- *UINT m_Kartyak_Talon:*
  A talonban levő kártyák értékei. (Több lap is lehet.)
- *UINT m_Kartyak_...:*
  A játékosok kártyáinak értékei (típusai). (Több lap is lehet.)
- *UINT m_Kezdo_Jatekos:*
  Melyik játékos kap több lapot leosztásnál. Ő kezdi a licitálást.
- *CString m_JatekosNevek[3]:*
  A játékosok nevei. 0-dik helyen a saját, 1-n a körben utánam következő, 2-on a harmadik játékos neve.
- *CVarakozas var:*
  A várakozás ablak egy példánya. Ezzel rajzoljuk ki és tüntetjük el.
- *CString m_Server_Address:*
  A szerver címe. (Csak kliensnél.)
- *CStatic *m_...[12]:*
  A játékosok kártyái. A *CStatic m_...* változókra mutatnak. Az egyszerűbb kezelhetőség miatt került bevezetésre.
- *CString m_Nev:*
  Saját nevem (program ezen példányát futtató játékosé). Az *ulti.cfg* fil-ból töltődik be.
- *UINT m_Port:*
  A port, amin kommunikál a szerver és a kliensek. Az *ulti.cfg* fil-ból töltődik be.

- *bool isServer:*
  Szervert futtatunk-e. Azért szükséges, mert a szerver és a kliens sok mindent másképp csinál, főleg az üzenetek feldolgozását. És vannak olyan típusú üzenetek, amelyeket csak az egyik kaphat meg, így a másik ezeket nem kezeli le.
- *int m_Connected:*
  Hány kliens csatlakozott már (csak szerver). Ez alapján tudjuk, hogy elindíthatjuk-e a játékot, illetve engedélyezzük-e a kliens csatlakozását (játék megtelt).
- *CClient* m_Client:*
  A kliens (csak kliens).
- *CClient* m_ClientSockets[2]:*
  A kliensekkel kommunikáló socketek (csak szerver).
- *CServer* m_Server:*
  A szerver (csak szerver).

Függvényei:

- *CUltiDlg(CWnd* pParent = NULL):*
  Konstruktor. Fejlesztőkörnyezet által generált.
- *void ExitGame():*
  Megszakítja a kapcsolatot a többi játékossal. Az ablak jobb felső sarkában lévő X, a Játék menü Kilépés vagy Szétkapcsolás pontjára kattintva hívódik meg, illetve ha valaki más kilépése miatt END üzenetet kapunk.
- *bool LoadSettings():*
  Betölti az *ulti.cfg* file tartalmát, és ezekkel inicializálja az *m_Nev* és *m_Port* változókat. Ha a művelet sikeres, *true*-t ad vissza, ha nem, *false*-ot. A program indításakor kerül meghívásra.
- *bool SaveSettings():*
  Menti az *m_Nev* és *m_Port* változók tartalmát az *ulti.cfg* file-ba. Ha a művelet sikeres, *true*-t ad vissza, ha nem, *false*-ot. Akkor kerül meghívásra, ha a betöltés sikertelen volt – ekkor alapértelmezett adatokat ír ki –, illetve akkor, amikor a Beállítások ablakban az OK-ra kattintunk.
- *void Send(CString &nev, UINT type, UINT toSend):*
  Üzenetet küld. A küldendő üzenet *m_Name* mezője *nev*, *m_Type* mezője *type*, *m_Message* mezője *toSend* adatokat veszi fel. Kliens esetén az üzenet a szervernek, szerver esetén az összes csatlakozott kliensnek megy.
- *bool StartServer(UINT port):*
  Elindítja a szervert, amely ezután a *port* porton fog figyelni. Ha a szervert nem lehet elindítani (pl. az a port már foglalt), hamis értékkel tér vissza.
- *void ConnectToServer(LPCSTR serverName, UINT port):*
  A klienst csatlakoztatja a *serverName* nevű szerverhez a *port* porton. Sikertelen csatlakozás esetén a megfelelő hibaüzenettel figyelmeztet erre.

- *void SendMsg(CClient *client, CMsg *msg):*
  Üzenetet küld csak a *client* kliensnek. Az üzenet az *msg* által mutatott értékeket fogja tartalmazni.
- *CMsg* ReadMsg(CClient *client):*
  Beolvassa a *client* kliens által küldött üzenetet, és visszatér egy az üzenetre mutató pointerrel. A *client* lehet egy a szerver által használt kapcsolattartásra használt adattag is, így a kliensek is ezzel a függvénnyel olvassák a szerver üzeneteit.
- *void ProcessPendingRead(CClient *client):*
  A *client* kliens által küldött üzenet(ek) feldolgozását végző függvény. A különböző üzeneteket típusuktól és attól függően dolgozza fel, hogy szervert vagy klienst futtatunk.
- *void ProcessPendingAccept():*
  Akkor hívódik meg, ha a szerverhez csatlakozott valaki. Ez dolgozza fel a játékba lépési kéréseket. Beengedi a klienst, ha még nem telt meg a játék (*m_Connected* < *3*), különben egy FULL üzenetet küld, és bonja a kapcsolatot.
- *CString Eredmeny():*
  Leosztás végén a rendelkezésre álló adatok (húszak, negyvenek, ütések, utolsó ütés, ki vitte az utolsó kör, ki játsza az aktuális játékot) alapján megállapítja a játszott játékok kimenetelét (teljesítette vagy elbukta), és ezt szövegesen visszaadja.
- *bool Uti(UINT mi, UINT mit):*
  Visszaadja, hogy *mi* üti-e *mit*. Azaz nagyobb lap vagy adu-e a *mit*.
- *UINT KiVitte(UINT kezdo):*
  A *kezdo* sorszámú játékos függvényében a szerver megállapítja, hogy az adott kört ki viszi, és ezt a tudását egy KEZD típusú üzenet formájában megosztja a kliensekkel.
- *int HuszErtek(UINT hn):*
  A paraméterként megadott húszak és negyven kódját tartalmazó nemnegatív egészből kiszámolja azok értékét és visszatér vele.
- *UINT HuszNegyven(UINT pakli):*
  Az adu színétől függően előállítja a *pakli* által tartalmazott húszak és negyvenek kódját, és azt adja vissza.
- *bool VanNagyobbKartya(UINT minel):*
  Megállapítja, hogy a kezemben lévő kártyák közül van-e *minel* nagyobb értékű, vele azonos színű lap.
- *bool IsTizes(UINT kartya):*
  Igazat ad vissza, ha *kartya* tetszőleges színű tízes. Különben hamisat.
- *bool NagyobbKartya(UINT minel, UINT mi):*
  Azonos színű kártyákról eldönti, hogy *mi* nagyobb-e *minel*.
- *UINT KartyaSzine(UINT kartya):*
  Visszatér a *kartya* színkódjával.

- *bool JoLap(UINT kartya):*
  Az asztalon lévő, a kezemben „tartott” lapoktól, és a játék típusától (színes, színtelen, adu színe) függően megállapítja, hogy a *kartyat* rakhatom-e vagy nem. (Színre-szín és kötelező felülütés szabályának ellenőrzése.)
- *void KartyakatMutatMind():*
  Kirajzolja az összes játékos kártyáját lappal felfelé. Terített játékoknál használatos. Hívása előtt szükséges, hogy minden résztvevő kezében lévő lapok ismertek legyenek a hívó alkalmazás számára.
- *CString GetSzin():*
  Az *m_Jatek_Szine* változó kifejezése szöveges alakban.
- *void ResetLicit():*
  A licit mezőt visszaállítja alapállapotba, azaz minden pipát kivesz a jelölőnégyzetekből és letiltja a Terített mezőt.
- *void UjLeosztas():*
  Mindent visszaállít ahhoz, hogy egy új leosztást lehessen kezdeni, átadja a kezdés jogát a következő játékosnak, majd a szerverrel megkeveretni a lapokat.
- *void StartGame():*
  Három passz „elhangzása” után hívódik meg. Beállít mindent, hogy a játékot el lehessen kezdeni, kiszámolja a húszakat, beállítja a játék színét (színes, nem piros játéknál megkérdezi az adu színét).
- *UINT KartyakSzama(UINT pakli):*
  Megmondja, hogy a *pakli*-ban hány darab kártya található.
- *void KartyatTesz(UINT hanyadik):*
  Ha még a licit folyik, és nálunk van a több lap, akkor ez kiveszi a *hanyadik* lapot, és lerakja talonba. Ha már játszunk, akkor az asztalra rakja le, feltéve ha a szabályoknak megfelelő lapot akarunk lerakni. Kör végén ez számoltatja ki, hogy ki vitte, játék végét ez kéri a kiértékelést.
- *UINT JatekErteke(UINT jatek):*
  Kiszámítja, hogy a *jatek* mennyit ér. Ezt a játék kódja és az előre definiált értékek alapján teszi.
- *CString PrintGame():*
  Szöveggé alakítja az *m_Jatek* változó tartalmát. Ez azért fontos, hogy az ember is tudja a másik mit akar játszani.
- *UINT SetGame():*
  A jelölőnégyzetek értéke alapján összeállítja a játék kódját, és ezzel tér vissza.
- *void Licital():*
  Ha az általunk kijelölt játék „többet ér”, mint az eddigi legnagyobb, akkor beállítja ezt helyette, és a többieknek is elküldi a talonnal együtt. Ezt is csak abban az esetben teszi meg, ha rajtunk a sor a licitálásban.
- *bool SzintelenJatek(UINT jatek):*
  Megállapítja a *jatek*-ról, hogy színtelen-e vagy sem. (Megjegyzés: piros betli és piros durchmars is színtelen, ezeket máshol rebetlinek illetve

redurchmarsnak nevezik. Ezeket a játékokat nem lehet színesben játszani, csak más – színes – játékkal kombinálva.)

- *void KartyakatMutatSajat():*
  Kirajzolja a kezemben levő lapokat, azaz az *m_Kartyak_Sajat* változó tartalmát megjeleníti grafikus formában. Mindig sorba rakva, balra rendezve.
- *UINT log2(UINT n):*
  Kettesalapú logaritmus. Csak kettő hatványokra ad helyes eredményt.
- *UINT KartyatKivesz(UINT pakli, UINT hanyadik):*
  Megállapítja, hogy a *pakli*-ban a *hanyadik* lap mi. Ez a visszatérési értéke. Ezt a játék „színtelenségétől" függően teszi.
- *void KartyakatMutat():*
  A játék kezdetén (licit után) kirajzolja a játékteret (saját lapok, ellenfeleké – hátlapok, talon, asztal).
- *void Kever():*
  Mindenkitől „elveszi" a lapokat, és véletlen generátor segítségével megkeveri őket. A generátor általa előállított számokat 0 és 31 közé transzformálja. Az első tíz különböző szám által reprezentált kártya a szerveré lesz, a második tíz az egyes ellenfélé, a harmadik tíz a kettesé. A maradék két lapot odaadja annak, aki a licitálást nyitja.
- *void NevekKiirasa():*
  Klienseknél alkalmazott funkció. Ha már megkapták az összes játékos nevét és helyzetét a szervertől POS típusú üzenet formájában, akkor ezek alapján feltöltik a saját *m_JatekoNevek* tömbjüket helyes értékekkel, és a neveket meg is jelenítik.
- *void ErtekSzamolas():*
  A licit csoportba tartozó jelölőnégyzetek alapján kitölti az *m_Ertek* és *m_Plussz* mezőket, és ezeket meg is jeleníti.
- *void InitCardArrays():*
  A fejlesztő környezet által a lapokhoz rendelt változók összefogása megfelelő csoportokba. Ezek a csoportok: saját lapok, ellenfelek lapjai, asztalon lévő kártyák. Ezen kívül ez végzi még a kártyák képeinek betöltését a *m_CardBitmaps* tömbbe.
- *BOOL OnInitDialog():*
  A főablak létrehozásakor fut le. Kezdeti beállításokat hajt végre, mint változók inicializálása beállítások betöltése.
- *void OnSysCommand(UINT nID, LPARAM lParam):*
  Az ablak bal felső sarkában levő ikonra kattintás esetén behozza a rendszer menüt.
- *void OnPaint():*
  Az ablak kirajzolását végzi, ha valami más eltakarta eddig, vagy explicit kérjük ezt.
- *void OnJatekKilepes():*
  A Játék menü Kilépés pontjára kattintva hívódik meg. Szükség esetén lezárja a hálózati kapcsolatokat, majd kilép a programból.

- *void OnSugoNevjegy():*
  A Súgó menü Névjegy pontját kiválasztva fut le. Egy ablakot nyit, amely kiírja a játék és a készítője nevét.
- *void OnJatekBeallitas():*
  A Játék menü Beállítások pontjának kiválasztásakor felhozza a Beállítások dialógusablakot.
- *void On20_100():*
  A 20-100 jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti.
- *void On40_100():*
  A 40-100 jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti.
- *void OnBetli():*
  A Betli jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti. Attól függően, hogy „beraktuk a pipát” vagy „kivettük” engedélyezi vagy letiltja a Terített mezőt.
- *void OnDuri():*
  A Durchmars jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti. Attól függően, hogy „beraktuk a pipát” vagy „kivettük” engedélyezi vagy letiltja a Terített mezőt.
- *void OnNegyasz():*
  A Négyász jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti.
- *void OnPassz():*
  A Passz jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti.
- *void OnPiros():*
  A Piros jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti.
- *void OnTeritett():*
  A Terített jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti.
- *void OnUlti():*
  Az Ulti jelölőnégyzetre kattintva hívódik meg. Beállítja a többi – ebbe a csoportba tartozó – négyzetet, az Érték és Plussz mezőket, majd mindezt megjeleníti.

- *void OnJatekUjjatekinditasa():*
  A Játék menü Új játék indítása menüpontjára kattintva fut le. Elindítja a szervert. Ha ez sikertelen, hibaüzenet ad.
- *void OnJatekCsatlakozas():*
  A Játék menü Csatlakozás pontjának kiválasztása esetén fut le. Előhozza a csatlakozás dialógusablakot, majd megpróbál csatlakozni az ott megadott című szerverhez.
- *void OnJatekSzetkapcsolas():*
  A Játék menü Szétkapcsolás pontját kiválasztva hívódik meg. Üzenetet küld a többieknek, hogy ő most kilép, és lezárja a hálózati kapcsolatokat.
- *void OnDestroy():*
  Az főablak bezárása (az ablak bal felső sarkában található X-re kattintva, rendszer menü Bezárás pontját kiválasztva vagy ALT+F4) esetén kapja meg a vezérlést. Lezárja a hálózati kapcsolatokat, majd kilép.
- *void OnAtrendez():*
  A saját kártyáink fölött található gomb lenyomásakor átrendezi a lapokat színes, illetve színtelen játéknak megfelelő sorrendben (tízes a király felett illetve alsó alatt).
- *void OnPasszButton():*
  A Passz gomb lenyomásakor hívódik meg. Ha még semmilyen licit nem hangzott el, akkor ez a Passz jelölőnégyzet kiválasztása és a Licit gomb lenyomásának felel meg. Különben passzol, azaz PASSZOL típusú üzenetet küld.
- *void OnLicit():*
  A Licit gomb lenyomása váltja ki. Leellenőrzi, hogy a kiválasztott licit megfelelő-e (rá lehet-e mondani az eddigi legnagyobbra), és ha igen, egy LICIT típusú üzenetben elküldi a többieknek.
- *void OnSajat...():*
  A saját lapjainkra kattintva ez a függvénycsoport kapja meg a vezérlést. A *KartyatTesz* függvényt hívják meg a saját sorszámukkal.
- *void OnTalon():*
  A talon valamely lapjára kattintva fut le. Ha a játékosnak joga van rá, hogy kérhesse a talont (ő következik), akkor a lapok bekerülnek a saját kártyái közé. Ez az épp lerakott lapok visszavételére is alkalmas, amíg nem nyomtuk meg a Licit gombot.

2.10. A CVarakozas *osztály*

Ez által az osztály által megvalósított ablak jelenik meg, amíg megfelelő számú kliens nem csatlakozott a szerverhez.

2.11. Az ertekek.h *file*

Ez a file különböző konstansokat definiál, amelyeket a program több helyen használ.

A következő csoportok találhatók itt:

- *Játékok értékei:*
  A program innen veszi, hogy a játszott játékok mennyit érnek. Ha valaki ezeket az értékeket máshogy tanulta, itt kell megváltoztatni, hogy az ő ismereteihez igazodjon (és utána újra kell fordítani a programot).
- *Játékok kódjai:*
  Az üzenetküldéshez és belső ábrázoláshoz használt kódok. Minden kód bináris alakja egy – a többitől eltérő – helyen tartalmaz egyest, így kombinált játékok is egy szóban küldhetők és tárolhatók.
- *Üzenet típusok:*
  - **NAME:**
    A kliensek bejelentkezéskor elküldik a nevüket. Ilyen üzentet csak a szerver kaphat. Az ilyen típusú üzenetek *m_Name* mezőivel tölti fel az *m_JatekosNevek* tömböt. Ha mindkét kliens bejelentkezett, a neveket elküldi nekik egy POS típusú üzenetben.
  - **CARD:**
    Kártya küldése. A játék során lerakott kártya értéke (típusa) kerül ilyen módon küldésre. Csak egy kártya küldhető így.
  - **LICIT:**
    Ez a típusú üzenet a Licit gombra kattintáskor kerül küldésre, feltéve, hogy a megfelelő játékos megfelelő értékű játékot választott.
  - **POS:**
    A szerver elküldi a klienseknek, hogy szerinte melyik játékos milyen pozíciót foglal el (ő a 0-dik). Ezt kliensek átszámolják a saját viszonyítási rendszerükbe, és feltöltik a saját *m_JatekosNevek* tömbjüket.
  - **FULL:**
    A kliens kapja, ha már nem fér be a játékba. Ekkor kilép.
  - **PAKLI:**
    Több kártya – egy-egy játékos kezében levő mennyiség – elküldésére alkalmas üzenet. Keverés után a szerver így „osztja le” a lapokat a klienseknek.
  - **TALON:**
    Ezzel az üzenettel a talon lehet letenni és felvenni, illetve a felvett kártyákat is ezen keresztül kapják meg a kliensek. Ha az *m_Message* tartalma 0, akkor letettük a kártyát, ha 1, akkor felvettük, ha ezeknél nagyobb, akkor a talon tartalmát (két lap) kapjuk/küldjük.
  - **END:**
    Kilépés, szétkapcsolás esetén küldjük, és akkor kapjuk, ha valaki más lépett ki/kapcsolt szét.

- **PASSZOL:**
  Azt jelzi, hogy a játékos nem akar nagyobb játékot mondani. Az ilyen üzenetek növelik az *m_Passzok* változó értékét.
- **ADU:**
  Adu színének tudatása a többiekkel.
- **TERIT:**
  Terített játékok első köre után küldi a szerver. Az üzenet tartalma az *m_Name*-hez tartozó játékos lapjai. (Addig ugyanis a kliensek nem ismerik az ellenfelek lapjait.)
- **KEZD:**
  A szerver miután kiszámolta, hogy az aktuális kört ki vitte, ezt így tudatja a kliensekkel.
- **VEGE:**
  Azt jelzi, hogy véget ért a leosztás és új következik. Szerver küldi.

- *Kártyalapok:*
  Minden kártyához rendel egy kódot, amely bináris alakjában pontosan egy helyen tartalmaz 1-est (és mindegyik különböző helyen). Mivel egy gépi szó 32 bites architektúrán 32 bit, ezért a magyar kártya lapjai pont belefférnek.
- *Színek:*
  A magyar kártyában található négy szín kódja. Alkalmas a megfelelő színű lapok kiválasztására egy pakliból.
- *Húszak és negyven:*
  A játékosoknál található húszak és negyven tárolásához szükséges kódok.

Melléklet

A készítő által ismert ulti lényegesebb szabályai (lehetséges eltérések más szokásoktól):

- Betli és durchmars önmagában sohasem színes, még akkor sem, ha piros. Ez csak azt jelenti, hogy kétszer akkor értékű játékot játszunk. Máshol hívják rebetlinek illetve redurchmarsnak.
- Egy licitre csak nála nagyobb értékűt lehet rámondani, azzal a kivétellel, hogy nem piros játékra pirosat is licitálhatunk. Az utli plussza az értékét nem növeli, csak arra jososítja fel, hogy azonos értékű nem ulti fölé kerüljön.
- A játékok értékei a következők:
  - Passz: 1
  - 40-100: 4
  - Négyász: 4
  - Ulti: 4+1
  - Betli: 5
  - Durchmars: 6
  - 20-100: 8

  Ezen játékok mindegyikének értéke a duplája pirosban. A terítés csak a belti és a durchmars értékét duplázza meg.

A játék részletesebb leírása megtalálható a következő weboldalon:
http://www.cs.elte.hu/~vuk/ulti.htm