Chatjáték

Programozás C++ nyelven

A program alapvetően a lokális hálózaton valósít meg TCP alapú kapcsolatot két peer között. A kapcsolaton keresztül chat üzenetek váltására van lehetőségünk. Ezen felül tartalmaz a program UDP felderítést, amely a lokális hálózat broadcast címére küld egy speciális üzenetet, melyre a peer-ek reagálnak, ezáltal láthatjuk, hogy kik elérhetőek.

A program 7 funkciót tartalmaz, ezek a következők:

1.. Chat

2..Hálózat felderítése

3..Legutóbbi felderítés eredménye

4..Csatlakozás peer-hez

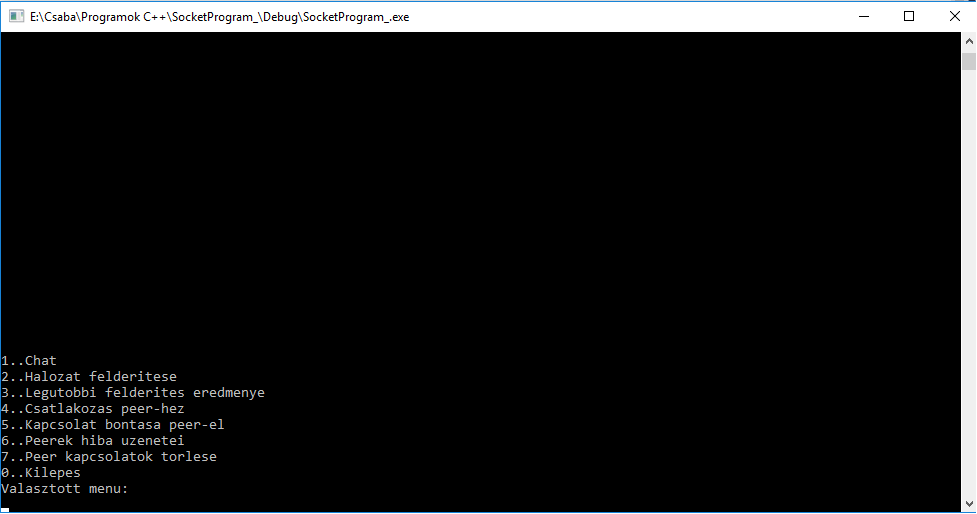
5..Kapcsolat bontása peer-el

6..Peerek hibaüzenetei

7..Peer kapcsolatok törlése

Továbbá a program tervezése során ügyeltem a bővíthetőség lehetőségére, ezáltal peer-ek között kivitelezhető külön féle játékok lebonyolítása.

Alap esetben a program mind TCP kapcsolatokra, mind UDP kapcsolatokra a 27015-ös portot használja, mely további funkciók hozzáadásával módosíthatóvá válik.



Kép a program fő menüjét ábrázolja, mely tartalmazza az említett hét funkciót.

Szerkezeti felépítés

ConnectionBase osztály: Betölti a kapcsolat létrehozásához szükséges állományokat, tartalmaz listát hibaüzenetek kezeléséhez, a hibaüzenetek száma maximálisan tíz, ha több hiba üzenet kerül a listára akkor az első elem törlésre kerül. Mivel a hiba üzenetek több szálból is érkezhetnek egy mutex objektum segítségével van védve a kritikus mező. Továbbá tartalmaz tisztán virtuális függvényt, tehát absztrakt osztálynak minősül.

UdpCom osztály: Leszármazottja az imént említett absztrakt osztálynak. Az osztály feladata, hogy megvalósítsa az udp felderítés. Feladatköre előre megadott üzenet küldése az általa automatikusan meghatározott broadcast címre, és a 27015-ös UDP porton várja más peer-ektől származó broadcast felderítést.

Server osztály: A ConnectionBase leszármazottja, feladata a TCP kapcsolatok fogadása a TCP 27015-ös porton. Beérkező kapcsolat esetén ’ConnectionAccepted’ eventet hív meg, mely tovább adja a fogadott kapcsolathoz tartozó socket-et. Gondoskodni kell róla, hogy az említett event-et egy másik objektum fogadja, mielőtt bejövő kapcsolat létrejöhetne.

Client osztály: A kliens osztály a ConnectionBase leszármazottja, és adatok küldésére és fogadására szolgál. Két alapvető konstruktort különböztetünk meg, az első által létrehozott kliens objektum kapcsolódhat másik peer-hez (ugye a másik peer, a server osztály segítségével fogadja a kapcsolatot), a másik konstruktor egy már bejövő kapcsolatból elfogadott Socket-et vár.

ClientManager osztály: A kliens objektumok tárolására és menedzselésére szolgál. Ha csatlakozni szeretnénk egy másik peer-hez akkor az osztály ’connectToPeer’ metódusát kell meghívnunk, mely létrehozza az objektumot és ha a kapcsolódás sikeres felveszi a kliens listába. Ezenkívül tartalmaz Server mutató mezőt, mert a Server osztály event hívása itt kerül elkapásra, elkapás esetén a létrejött klienst is felveszi a listába. Kliens osztályokat csak ezen menedzser osztályon keresztül érhetünk el, tehát üzenet küldéshez tartalmaz metódusokat, és tárolja a kliensek fogadó szálait (amelyeken keresztül olvassuk a csatlakozott peer-ek üzeneteit).

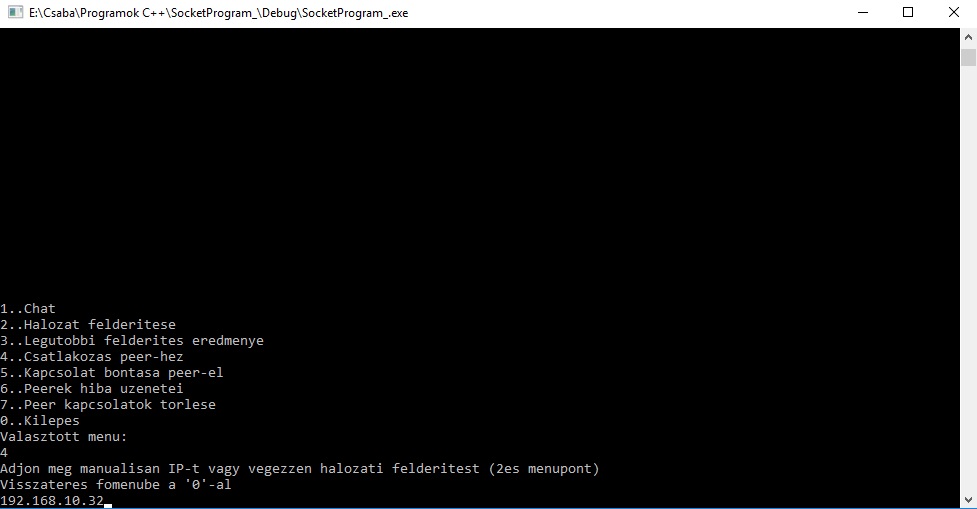
ClientMessages osztály: Minden kliens objektumhoz létre kell, hogy jöjjön, egy ilyen objetkum. Az osztály feladata egy lista menedzselése mely az üzeneteket tartalmazza. Mivel nem a ClientManager-ben tároljuk ezeket az adatokat, az azonosításhoz egy string objetkumot tartalmaz a peer ip címével, tehát, hogy ezek kinek az üzenetei.

CommData osztály: A kommunikációban létrejött adatokat kezeli. Egy ilyen objektumon kezeljük az adatokat, az osztályok melyek adatot közvetítenek, tartalmazzák ezen objektum referenciáját. Tárolja a felderített ip címeket, és ClientMessages-ből listát. A többszálból érkező üzeneteket felvételét a listába mutex objektumokkal védi. A két listához két külön mutex.

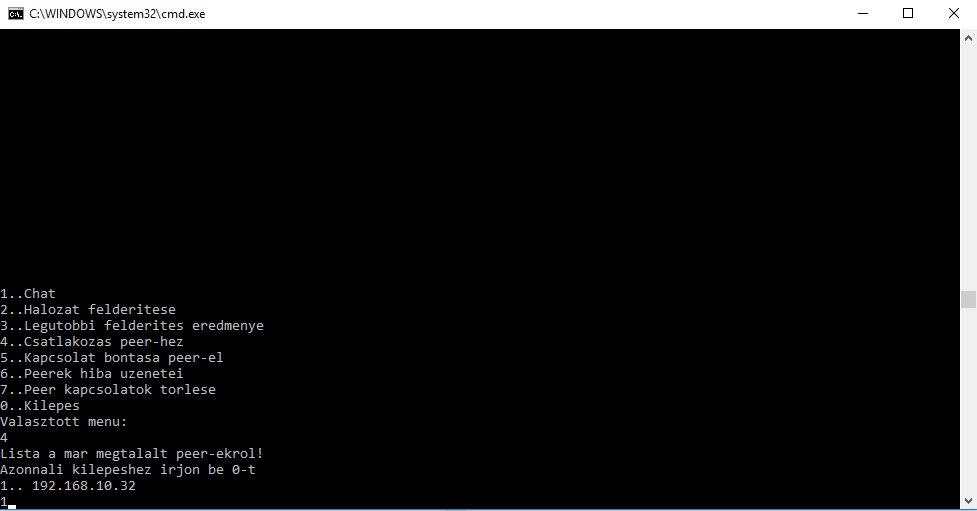
IOHandling osztály: Alapvetően egy egyszerű osztály mely a standard kimenetre ír ki dolgokat, illetve az onnan bekért adatokat ellenőrzi. Például itt van definiálva a metódus, ami ellenőrzi, hogy helyes ip címet adunk-e meg, amikor peer-hez csatlakozunk.

Control osztály: A vezérlő osztály, minden eddig említett osztály itt fut össze. A main függvényt tartalmazó Source.cpp fájl csak a Control.hpp-t tölti be, hisz az már mindent tartalmaz amire szüksége van. A fő menü kiválasztott pontjait vezeti le. Az összes ponthoz tartozik egy függvény, mely kezeli a teendőket az adott pontal kapcsolatban.

Például a 4. pont „Csatlakozás peer-hez” esetében ellenőrzi a CommData ip listáját, hogy tartalmaz-e elemeket, ha nem akkor tanácsot ad, hogy végezzen előbb hálózati felderítést, de manuálisan is megadhatunk címet. Ha a lista tartalmaz elemeket, akkor megjeleníti azokat, és a listából választhatunk címet a csatlakozáshoz. A metódus végén információt kapunk róla, hogy sikerült-e a csatlakozás.



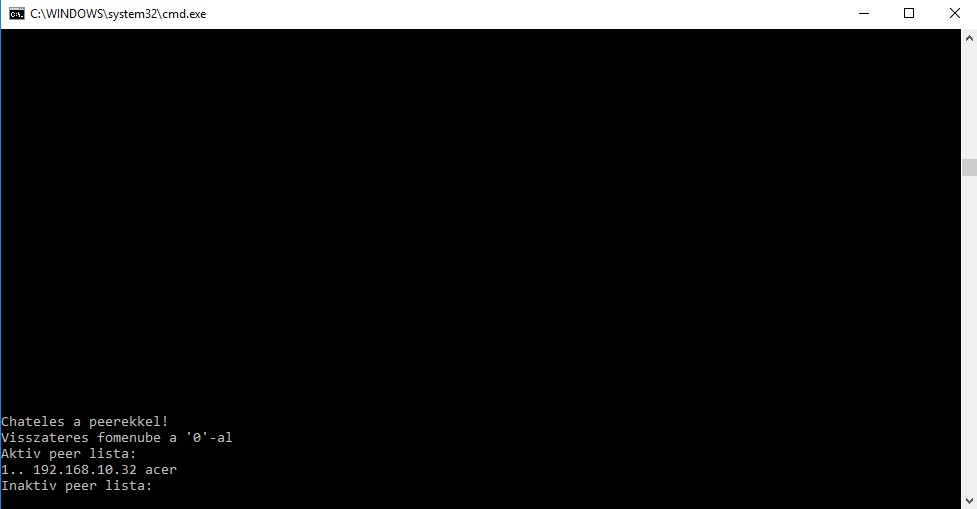
Első eset, nincs ismert IP cím.



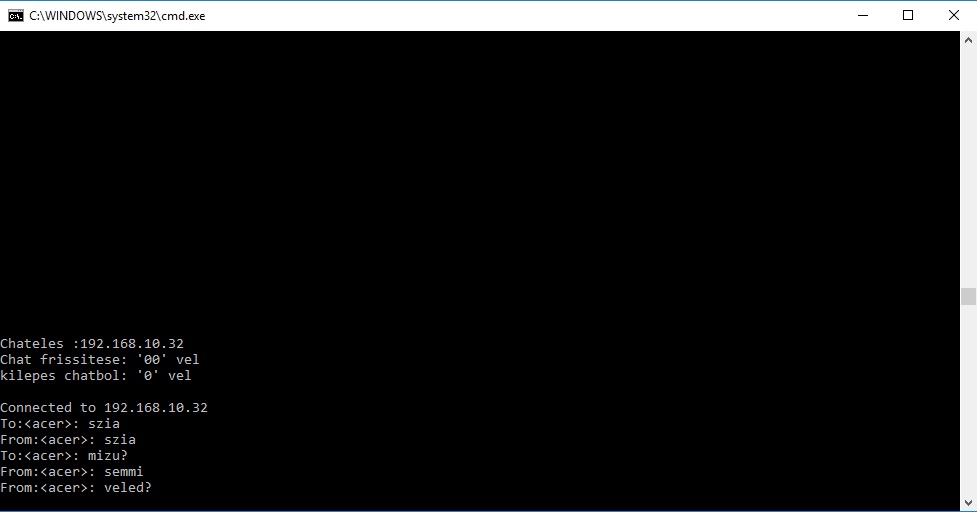
Második eset, listából választhatunk elemet.

Fő funkció chat bemutatása:

Chat használatához szükségünk van peer-ek ismeretére. Ha a menüben kiválasztjuk az 1-es opciót akkor először kapunk egy listát, hogy kinek szeretnénk üzenni. A lista két részre van osztva felül az aktív peer lista, akiknek üzenhetünk, alul az inaktív, akiknek üzenni nem tudunk, de a hiba üzeneteket megtekinthetjük velük kapcsolatban. Ha kiválasztunk egy peer-t már chatelhetünk is. Két különleges kód van melyet nem küldhetünk üzenetben, a ’0’ mely visszavisz a fő menübe, és a ’00’, mely frissíti a chatet, tehát van-e új bejövő üzenet.



A képen kiválasztottuk az 1-es opciót a menüben, és láthatjuk az aktív listát.



Részlet egy rövid chatelésből.