



M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

Nemes Tamás, Puspán Sára, Straubinger Dániel –
23-as csoport

BEÁGYAZOTT RENDSZEREK SZOFTVERTECHNOLÓGIÁJA – HÁZI FELADAT SPECIFIKÁCIÓ

KONZULENS

Erdős Csanád

BUDAPEST, 2017

1.1 Játékmenet	3
1.2 Felhasználói felület	4
2 Megvalósítás	5
2.1 UML diagram és főbb működési mód ismertetése	5

1 Feladatkiírás

A tantárgy keretében csoportunk feladata egy Rizikó játék megtervezése és implementálása JAVA nyelven.

1.1 Játékmenet

RIZIKÓ klasszikus stratégiai társasjátékban a 2-3 játékos játssza, céljuk a területek elfoglalása. A játék egy egyedi térképen játszódik. A területek védelme csak figyelmes tervezés után lehet sikeres. A súlyos vereségek fontos körzetek, vagy akár kontinensek elvesztését jelentik. A győzelemhez a merész csapatmozgásokon és gyakorlaton kívül a kockadobás szerencséje is szükséges. A Rizikó játékban a feladat az egész világ meghódítása.

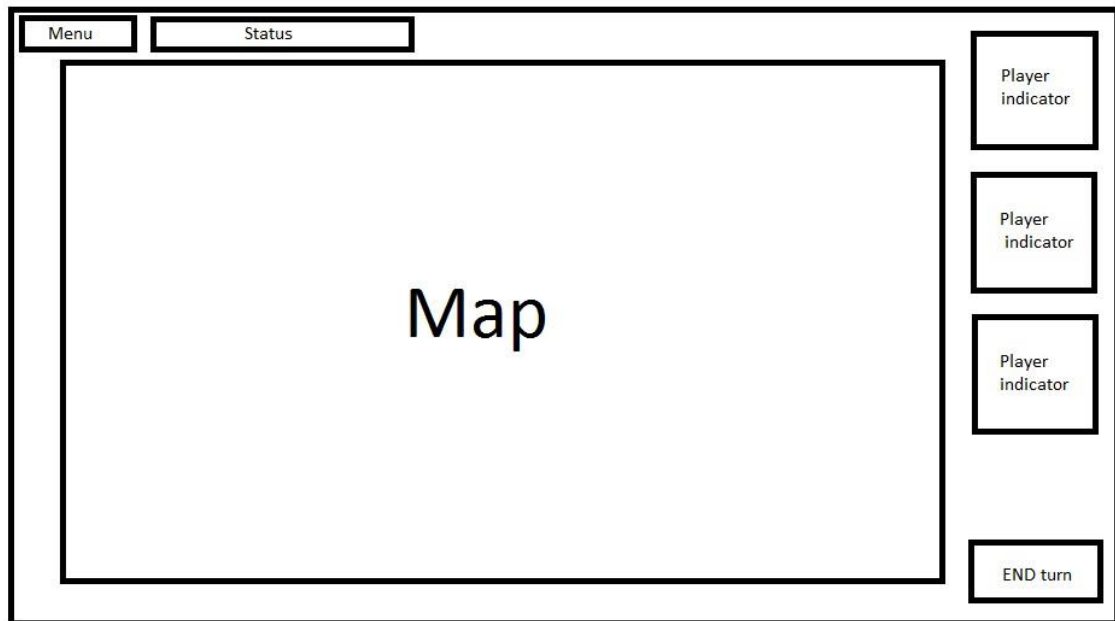
A játék körökre osztott. Az aktuálisan soron levő játékos az elfoglalt területek alapján a kör megkezdésekor új egységekkel gyarapodik. A soron következő játékos az általa elfoglalni kívánt terület megszerzéséhez támadást indít. A támadás során a támadó maximum három dobókockával (egységek számától függően), a védekező pedig kettővel dob. Támadást csak szomszédosan kapcsolódó országok között lehet megindítani. Ha a védekező legalább egyenlőt dob, akkor a támadó, ellenkező esetben a védekező veszít egységet (például a támadó 6-ot, 4-et illetve 3-mat dob, a védekező pedig két darab 5-öst, akkor mindketten egy-egy egységet veszítenek) – végül a védekező egységek elfogyása esetén az adott ország a támadóhoz kerül, ahova legalább egy embert át kell csoportosítani (tehát a támadás megkezdéséhez és befejezéséhez legalább két emberrel kell, hogy rendelkezzen).

A játék elején a játékosok között kezdő kontinensek véletlenszerűen kerülnek kisorsolásra, majd a kezdeti csapatokat ezek között szabadon oszthatják el. Az adott körön levő játékos választja ki, hogy milyen műveleteket kíván végezni, illetve támadás esetén a védekező is becsatlakozik a körbe: mindketten gombnyomásra véletlenszerűen dobhatnak a kockákkal. A játékos a körének végét gombnyomással jelzi, ezután automatikusan a következő játékos kerül sorra.

A játék automatikusan véget ér, ha egy játékos elfoglalja az egész térképet, illetve egy adott játékos számára véget ér, hogyha az összes egységét elveszítette.

1.2 Felhasználói felület

Az alábbi 1. ábrán látható a GUI vázlatos terve.



1. ábra - GUI vázlatos megjelenítése

1.2.1 ábra - GUI vázlatos megjelenése

A menüben lehetőség nyílik új játék kezdetére, az aktuális állás elmentésére (illetve korábbi játékmenet folytatására), kilépésre, valamint újraindításra. A játékosok legfontosabb adatai a felhasználói felület jobb oldali részén jelenítődnek meg, illetve a kör befejezésére a jobb alsó sarokban levő gombbal lesz lehetőség.

Támadás esetén egy felugró ablakban zajlik le az ütközet, ahol a támadó és megtámadott félnek egyaránt lehetősége nyílik az ütközet lefolytatására „kockadobás” útján. A soron következő játékos a térképen (map) egy adott országot kiválasztva egy felugró ablakban választhat, hogy megtámadja-e, illetve a támadás menetközben is félbehagyható (például a támadó úgy dönt, hogy már túl sok egységet veszített).

A térkép felépítése olyan, hogy nem minden ország kapcsolódik egymáshoz (egységmozgás csak szomszédos területek között lehetséges). Több ország egy nagyobb egységet, kontinenst, testesít meg, aminek a teljes elfoglalása esetén a kör eleji bónusz egységek száma is növekszik.

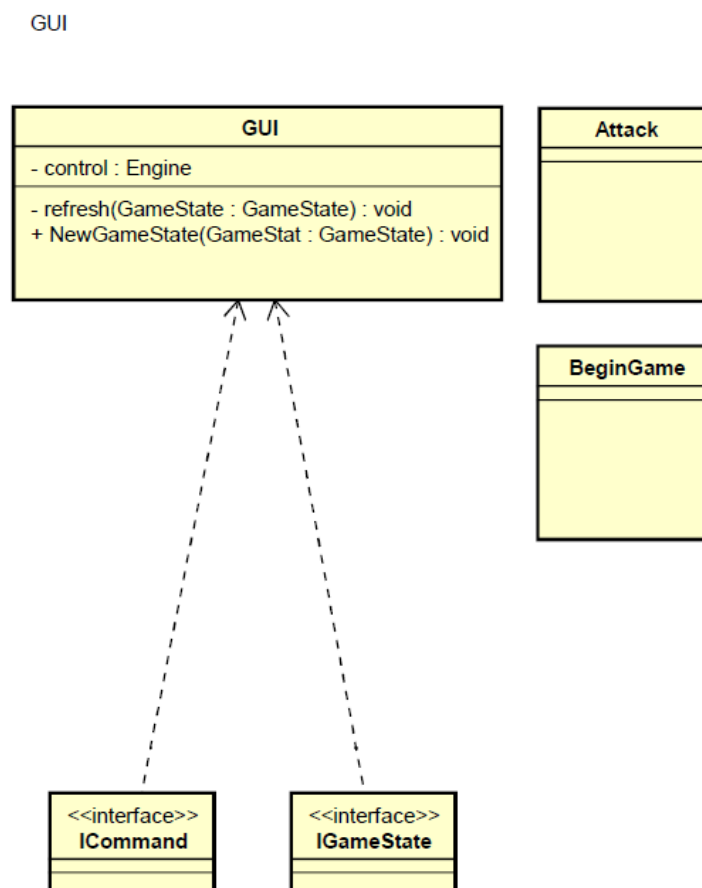
2 Megvalósítás

2.1 UML diagram és főbb működési mód ismertetése

2.1.1 GUI – Grafikus felhasználói felület

Az UML diagramokokat a csoport közös tervezés után, az Astah Professional program segítségével valósította meg.

Ahogy a feladatot, úgy az UML diagramot is három fő részre bontottuk: GUI-ra, Engine-re és Networkre. Az következőkben ismertetjük az UML diagramot, valamint a megvalósítandó főbb funkciókat. A megvalósításhoz főleg Swing komponensek kerülnek majd felhasználásra.



2. ábra – GUI UML

A teljes program UML diagramja az utolsó oldalon, az 6. ábrán látható. A grafikus felhasználó felület az Control résszel lesz kapcsolatban. Felépítését tekintve három fő rész található meg benne: GUI, NewPlayer, Attackscreen.

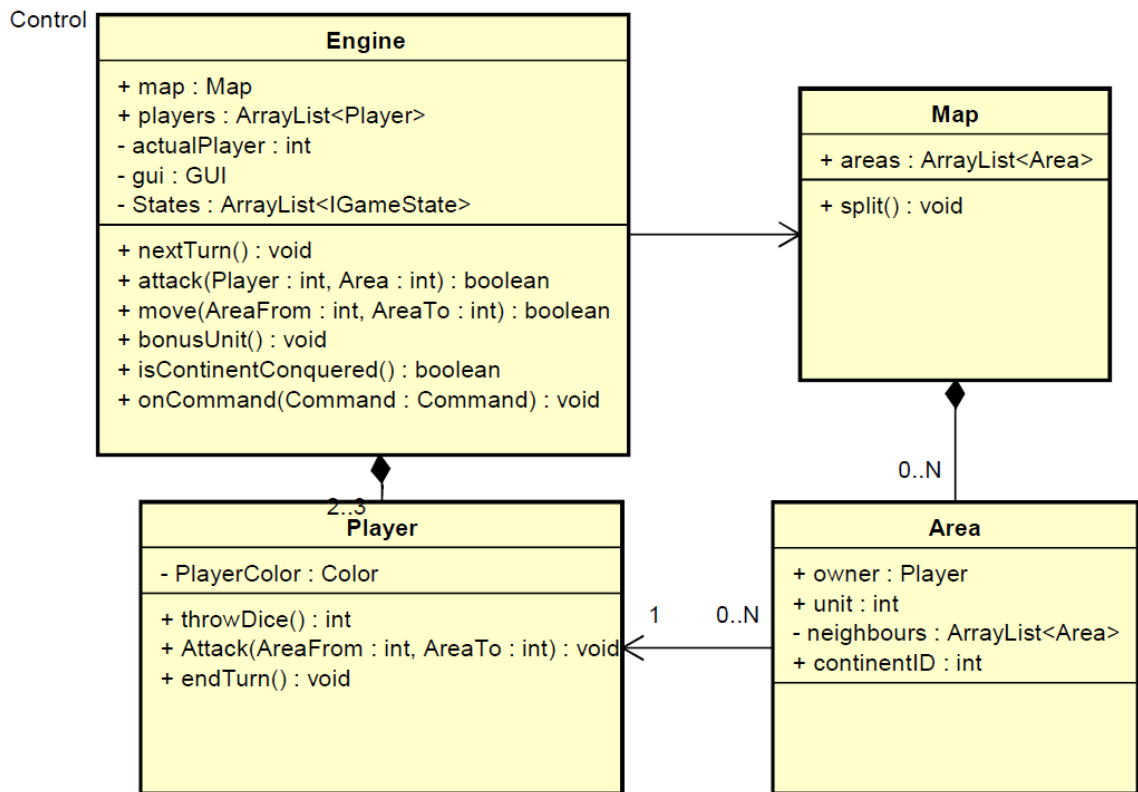
A GUI-ban kerül megvalósításra kvázi az 1. ábrán bemutatott felület: tartalmazni fogja a kattintható térképet, egy menüsor, az egyes játékosok státuszát, valamint egy kör vége gombot. Megvalósítás tekintetében a térkép egy alfa csatornával rendelkező PNG formátumú képfájl lesz. Az egyes országok pozíciójának meghatározása MouseListenek segítségével kerül megvalósításra, a kép fölött elhelyezett JLabel-ök felhasználásával. A menüsorból indítható lesz a másik két JFrame, a NewPlayer és az AttackScreen.

A NewPlayer, ahogy a neve is mutatja, új játékos hozzáadására szolgál. A megjelenő ablakban a játékosnak meg kell adnia a nevét, és a kiválasztott szint, amivel azonosításra kerül majd a térképen.

Az AttackScreen felületen valósul meg a csata két játékos (a támadó és a támadott) között. A felületen feltüntetésre kerülnek a játékosok adatai, valamint a kockadobásra is itt kerül sor, ami alapján a csata győztese kiszámításra kerül.

2.1.2 Control - irányítás

A felhasználói felületen megvalósított feladatok az Control részben megírt függvények segítségével történnek. Ennek UML diagramon való megvalósítása a 3. ábrán látható.



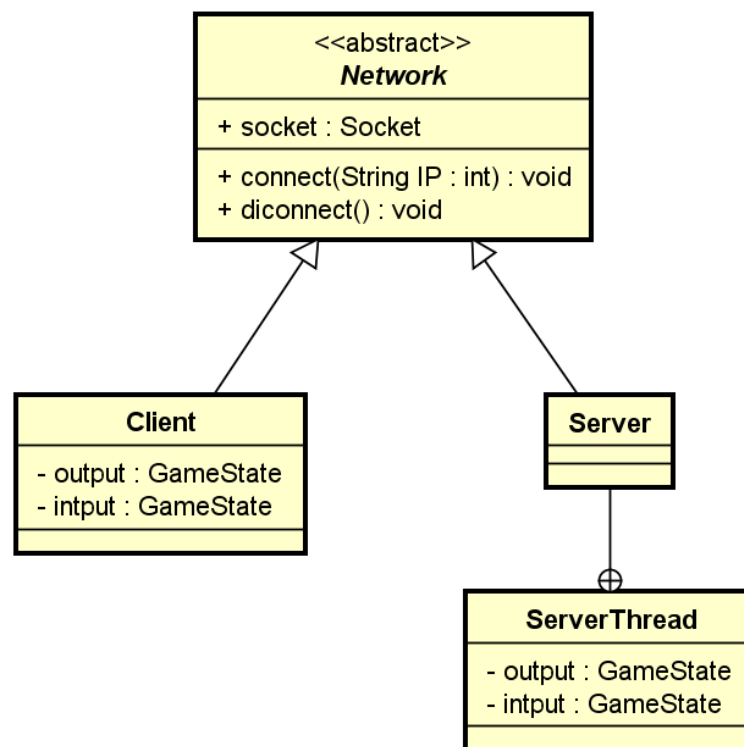
3. ábra - Control UML diagram

A Control egy Engine, Map, Player illetve Area osztályokból épül fel. Az egyes osztályok tartalmazzák az adott objektumokhoz szükséges feladatok megvalósítását. Ilyenek például a játékmenet rész alatt ismertetett folyamatok, például az az egységek mozgatása területek között (Move), másik játékos területének megtámadása (Attack), kockadobás véletlengenerátor használatával (throwDice), kör befejezése (endTurn). A játékosok és a területek egyaránt listákban kerülnek tárolásra. A területeknél szintén egy listában kerül tárolásra az egyes területek közötti szomszédossági viszony, ami alapján eldönthető, hogy adott területről támadható-e egy másik, illetve mozgathatóak-e egységek egy adott úton.

Minden egységhez tartoznak olyan változók és függvények, amik segítségével a felületen elvégezhető lesz az egyes játékosok nyomon követése (melyik terület kihez tartozik, kinek mennyi bónusz egysége van, a játékos mikor fejezi be a kört, birtokában van-e egy adott kontinensnek). A térkép (Map) tartalmazza a játék kezdetéhez szükséges, a területek játékosok közötti véletlen felosztásának függvényét. Minden játékoshoz (Player) tartozik egy szín is (PlayerColor), ami alapján grafikusan megkülönböztethetőek a felületen.

2.1.3 Network - hálózat

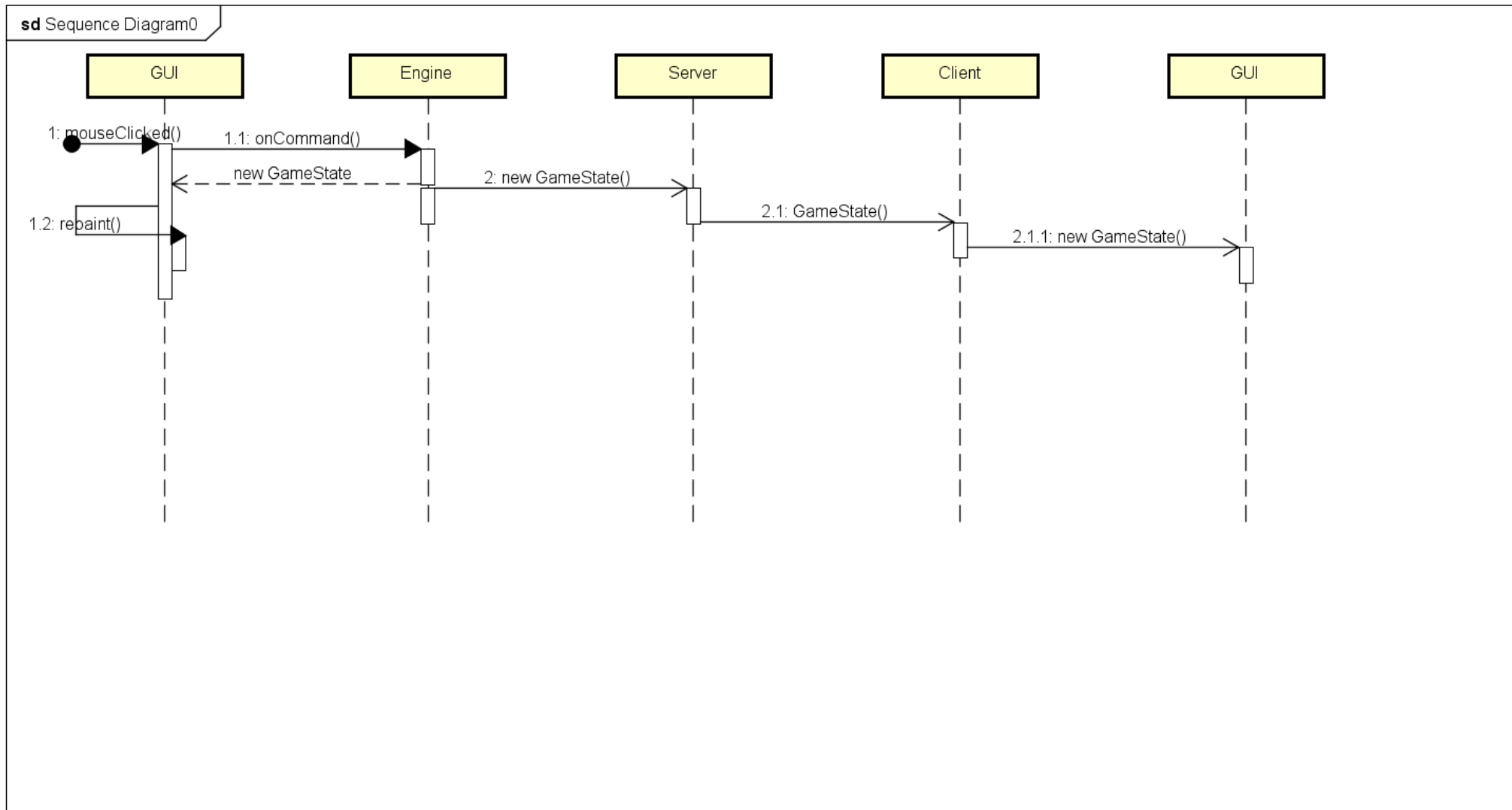
Mivel a játékot több személy játssza, ezért szükség van a játékosok közötti kommunikációra. A harmadik külön egységként és feladatként megvalósított rész a hálózat, ami a játékosok közötti kapcsolatot valósítja meg. Ennek leendőbeli megvalósítása a 4. ábrán szereplő UML diagramon található.



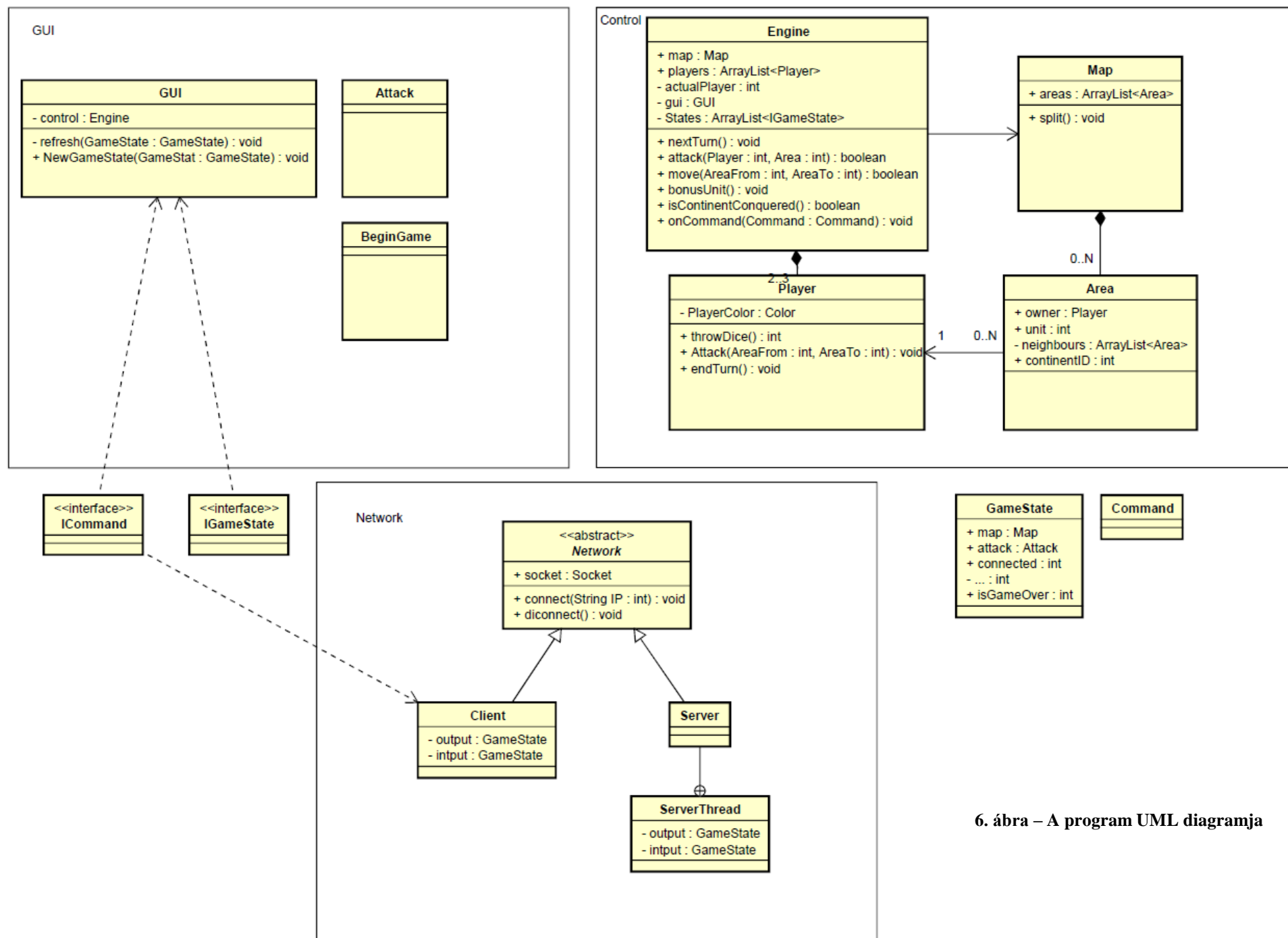
4. ábra - Network UML diagram

Az a rész valósítja meg a socketkezelést, a szerver-kliens kapcsolatokat. A játékosok, mint kliensek, egy szerverhez fognak kapcsolódni. A kommunikáció a GUI-val egy GameState interfész segítségével fog megvalósulni, amiben a GUI számára minden releváns információ tárolásra kerül.

Az egyes részegységek közötti kommunikáció szemlélteti az alábbi szekvencia diagram az 5. ábrán.



5. ábra - Szekvencia diagram



6. ábra – A program UML diagramja