# Rizikó játék fejlesztői dokumentáció

Készítette: Nemes Tamás, Puspán Sára, Straubinger Dániel

# GUI

## Térkép

A felhasználó felületen levő térkép egy statikus png formátumú képből áll, ami egy JLabel ikonjának van beállítva. Az egérmozgás illetve a területek érzékelése úgy történik, hogy minden egyes ország fölött JLabel-ök vannak elhelyezve. Ezek lefedik az egész térképet, és egy közös Hashmap-hez vannak hozzáadva, amihez egy actionListener van definiálva. Minden label kulcsa az adott ország neve, amivel a Motor territories vektorában szerepel.

Ezeken a labelökön felül minden ország közepén pluszban van mégegy JLabel, aminek színe automatikusan beállításra kerül annak a játékosnak a színére, akié az adott terület, valamint a közepében levő szám jelenti azt, hogy hány egysége van az adott területen a játékosnak.

Az egyek országokon levő indikátor körökön levő kattintással lehet új interakciót kezdeményezni: egységet mozgatni saját területek között, bónusz egységet lehelyezni saját területre (dupla kattintás az adott területre), vagy támadást kezdeményezni (akkor, hogyha a második kattintással kiválasztott terület nem az adott játékosé).

Adott terület fölé mozgatva a kurzort, annak képe és neve megjelenik a képernyő bal alsó sarkában. Ezt az egyes országokról elkészített png formátumú képek teszik lehetővé.

A menüsorból elérhető az új játékos hozzáadása funkció.

## Új játékos hozzáadása

Új játékos hozzáadásánál meg kell adnia a felhasználónak: játékos nevét, színét, valamint, hogy kliens vagy szerver szeretne lenni. Ha már létre van hozva egy szerver, akkor a kliens automatikus csatlakozik hozzá.

## Támadás

Támadás esetén automatikusan megjelenik a támadási képernyő, ahol a támadó fél a támadás gombot megnyomva véletlenszerű kockadobásokat eszközöl. Ha ez megtörtént, akkor megjelenik a képernyő a védekező oldalon, ahol a védekezést követően az eredmény visszakerül az támadó ablakába.

## Egyéb menüsorok

Az adott interakciók esetén megjelennek további menüpontok az egységmozgatáshoz illetve az egység lehelyezéshez. Ezek hasonló felépítésűek: az aktuális egységszámot jelzik, valamint a „+” illetve „-” gomb segítségével növelhető a kiválasztott egységek száma, ameddig lehetőség van rá.

# Motor

## Motor osztály

A motor legfontosabb részét a játékosok és a területek tárolását szolgáló Territory és Player típus, valamint az egyes elemeket felfűzve létrejövő territories és players vektorok alkotják.

A motorban kerültek megvalósításra azok a függvények, amivel a GUI-n levő interakciók megvalósítódnak: egységmozgatás, támadás, egység lehelyezés.

A motor függvényei meghívásra kerülnek a GUI-ban, támadási képernyőn, elvégezve a szükséges kalkulációkat. A motor felhasználja a GUI-n kattintással kiválasztott területeket, ami alapján eldönthető, hogy mi is az adott művelet.

A motoron belül meg van valósítva továbbá, hogy a támadás során dobott eredmények függvényében változzon az adott területek egységszáma, vagy pedig akár el is foglalódjon az adott terület a támadó kellő számú győzelme után (upDateUnitsAfterAttack).

Az assignPlayer() függvény segítségével a területek tulajdonosai véletlenszerűen kerülnek kisorsolásra.

Az assignAdjacentsandContinent() függvényben kerülnek beállításra az egyes területek szomszédossági listái.

Ezen felül kapcsolatot teremt a grafikus felület és kommunikáció között.

## Player, Territory osztályok

A Player és Territory osztályokban amikből a motor vektorban tárolja a példányokat, definiálva vannak az egyes objektumok szükséges tagváltozók.

Játékos (Player) esetében ezek rendre: név, index, elfoglalt területek, seregek, szín. A felhasználó színe alapján kerülnek beállításra a grafikus felület elemei (indikátorok színei).

Terület (Territory) esetében név, játékos (akihez tartozik), seregek száma, continens (amelyiknek része), szomszédok. A szomszédok tárolása azért fontos, mert támadás és mozgatás egyaránt csak akkor lehetséges két terület között, ha azok szomszédosak.

# Szerver kommunikáció

A két instance közötti kommunikáció TCP alapú. A szerver – amennyiben nem használta – megnyitva a 7555-ös socketet, amelyeln keresztül tudja fogadni a kliens féltől az adatokat.

A kommunikációt megvalósítja egy Net nevű abstract osztály, melyből a Client és a Server leszármazik. A közös függvények, Network mindketten *override*-olnak a sendGameState és a connect és a disconnect. A connect és a disconnect végzi az össze illetve szétkapcsolást. A Server és a Client külön szálon futnak.

## GameState osztály

A Server és a Client közötti kommunikáció egyetlen egy (Serializable) osztály segítségével zajlik. Az osztálynak publikus változó vannak, viszont csak lokálisan egy-egy függvényben hozunk csak létre. A GameState osztály tartalmaz minden információt, mely a két isntance szinkronizálódásához szükséges. Tartalmazza a területeket információját (*Vector<Territory>*), az egyes játékosokról szóló információkat (*Vector<Player> players*), valamint azt, hogy milyen állapotban van az aktuális játék (*state*).

A GameState osztály az alábbi öt esetben kerül létrehozásra:

1. Kör vége gomb megnyomása. Ilyenkor frissíti a területeket, valamint átállítja az aktuális játékost, illetve frissíti a játékoshoz tartozó területek listáját
2. Ha megtámadták az adott játékost: ilyenkor a támadó elküldi a védekező félnek a dobása eredményét, és a védekező előtt felugrik az ablak megjeleik, melyben figyelmeztetik, hogy megtámadták, és lehetősége van dobni
3. A védekező elküldi a saját eredményeit a támadónak
4. A támadó játékos befejezi a támadást, és az eredmény kiértékelésre kerül
5. Szinkronizáció: A kliens elküldi az információját a szervernek, és a szerver a kliensnek. A játékosnév és a szín kiemelt fontosságú

## GameState feldolgozása

A GameState osztály feldolgozását a *Motor* függvényei végzik. Minden egyes esthez külön kiértékelés tartozik, mely frissíti a saját adatait a kapott adatok alapján.