



EÖTVÖS LORÁND TUDOMÁNYEGYETEM

Informatikai Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

Trónfoglaló, stratégiai kártyajáték

Kitlei Róbert

Kubisch Dániel

Mestertanár

Programtervező informatikus

Budapest, 2018

Tartalomjegyzék

| | |
|---|-----------|
| 1. Bevezetés | 2 |
| 1.1 Témaválasztás oka | 2 |
| 1.2 A játék rövid leírása..... | 2 |
| 2. Felhasználói dokumentáció | 3 |
| 2.1 Rendszerkövetelmény | 3 |
| 2.2 A játék leírása..... | 3 |
| 2.3 Játékmenet..... | 4 |
| 2.4 Használati útmutató | 4 |
| 2.4 Kártyák..... | 8 |
| 3. Fejlesztői dokumentáció..... | 11 |
| 3.1 A játék részletes leírása..... | 11 |
| 3.2 A mappaszerkezet felépítése..... | 13 |
| 3.3 A program szerkezete..... | 14 |
| 3.4 Szerver – Kliens..... | 16 |
| 3.5 A játék logikája..... | 19 |
| 3.6 Adatbázisok és felépítésük | 22 |
| 3.7 Tesztelés tervé és eredmények..... | 30 |
| 4. Felhasznált irodalom | 38 |

1. Bevezetés

1.1 Témaválasztás oka

A témámat azért választottam, mert olyannal akartam foglalkozni, ami érdekel. Számos játékot kipróbálva, én is szerettem volna egy szórakoztató alkalmazást készíteni.

A szakdolgozatom témáját elsősorban a The Witcher 3: Wild Hunt ^[1] játékban található kártyajáték, a Gwent ^[2] ihlette.

A játék alapjai hasonlóak a Gwent kártyajátékhöz, a szabályok szinte megegyeznek, viszont a kártyalapokban vannak eltérések, valamint a program lehetővé teszi, hogy két ember játsszon egymás ellen. Az eredeti Gwent játékban az ellenség nehézségeivel és lépéseiivel is elégedetlen voltam, ezen szerettem volna javítani, így lehetővé vált, hogy egy logikus döntéseket hozó ellenfelet hozzak létre.

1.2 A játék rövid leírása

A Trónfoglaló egy stratégiai kártyajáték, egy- illetve többjátékos móddal. A játék két személyes, mindenki játékos egy általuk összerakott legalább 20 kártyából álló pakliból 15-15 véletlenszerű lapot húz fel. A kártyák közel- és távolharci egységek, időjárást változtató, speciális képességű lapok lehetnek. Mindkét játékosnak kettő élete van, akinek előbb elfogy az élete, veszít, így a játék 2 vagy 3 fordulós lehet. Döntetlen játék esetén újrajátszás következik.

Egyjátékos módban a felhasználó a számítógép ellen játszhat. Itt a játék célja egy térkép összes területének elfoglalása. minden elfoglalt terület után a játékos új lapokat kap, melyet elhelyezhet a paklijába, vereség esetén egy véletlenszerű lapot elveszít. Ha a játékosnak nem marad elegendő lapja, akkor veszít. Az itt megszerzett kártyákat többjátékos módban is használhatják.

Többjátékos módban két ember játszhat egymás ellen, itt nincsen térkép, és a két játékos egy partit játszik le. A nyertes nem kap, a vesztes nem veszti el egyik lapját sem.

2. Felhasználói dokumentáció

2.1 Rendszerkövetelmény

A játék futtatásához egy Java ^[3] futtatására alkalmas asztali számítógépre van szükség, valamint ajánlott egy legalább HD felbontású (1280x720) monitor.

2.2 A játék leírása

A Trónfoglaló játék két személyes, mindkettő játékos egy általuk összerakott legalább 20 kártyából álló pakliból 15-15 véletlenszerű lapot húz fel. A kártyák közel- és távolharci egységek, időjárást változtató, illetve speciális képességű lapok lehetnek. Az egység lapoknak lehet képességük, amellyel egy sor erejét növelik vagy csökkentik. Az időjárást változtató lapok, a Kód és a Fagy, a távolharci és közelharci lapok erejét csökkentik mindenki játékosnál. A Járvány lap a lerakott lapok közül törli a legkisebb értékűeket mindenki játékosnál. A fordulót az a játékos nyeri meg, akinek a lapjai összértéke a nagyobb, a vesztes elveszíti egy életét. Mindkét játékosnak kettő élete van, akinek előbb elfogy az élete, veszít, így a játék 2 vagy 3 fordulós lehet. Döntetlen játék esetén újrajátszás következik.

Egyjátékos módban a felhasználó a számítógép ellen játszhat, egy választott nehézségi szinten. A nehézségi szint lehet könnyű, közepes, illetve nehéz. A szintek között a különbséget a gép által használt taktikák jelentik. Könnyű nehézségi szinten a legegyszerűbb taktikákat használja, a nehéz szinten legösszetettebb taktikákat. A játék célja egy térkép összes területének elfoglalása. minden elfoglalt terület után a játékos egy új lapot kap, melyet elhelyezhet a paklijába. A paklit a főmenüből elérhető Edit Deck almenüben szerkesztheti a játékos. minden vereség után egy véletlenszerű lapot elveszít. Ha a játékosnak nem marad elegendő lapja, akkor veszít. minden lejátszott játék után a jelenlegi állás mentésre kerül, így a legközelebbi indításnál a játékot onnan lehet folytatni, ahol abbahagyta, vagy kezdhet új játékot. Az utóbbi esetben az eddigi mentések törlésre kerülnek. Az itt megszerzett lapokat többjátékos módban is használhatják.

Többjátékos módban két ember játszhat egymás ellen. Az egyik játékosnak el kell indítani egy szervert a Start Multiplayer gombra kattintva, utána a másik

játékos csatlakozhat a Join Multiplayer gomb megnyomásával. Ebben a módban nincsen térkép, a két játékos egy partit játszik le. A nyertes nem kap új lapot, a vesztes nem veszti el egyik lapját sem.

2.3 Játékmenet

Egy játék kezdésekor 15 kártyát kap a játékos, ha rajta van a sor lerakhatja egy lapját, vagy passzolhat. Az utóbbi esetben több lapot nem rakhatal a forduló végéig. Mindkét játékosnak kettő élete van, aki előbb elveszti minden kettőt az veszít.

A kártyákat bármilyen sorrendben le lehet rakni. Az értékük összeadódik. Amelyik játékosnak a két sorának az összértéke nagyobb, az nyer, döntetlen esetén minden két játékos életet veszít.

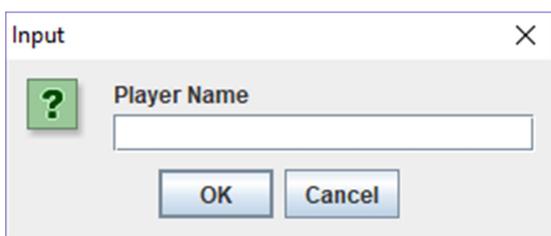
2.4 Használati útmutató

A játék indítása

A játékot a Tronfoglalo.jar állománnal lehet elindítani. Az indítóval egy mappában kell lenni a cards, data és lib mappáknak. Ezekben tároljuk a képeket, adatbázisokat és szükséges állományokat.

Bejelentkezés

A játék indításakor a bejelentkező ablak jelenik meg (1. ábra).

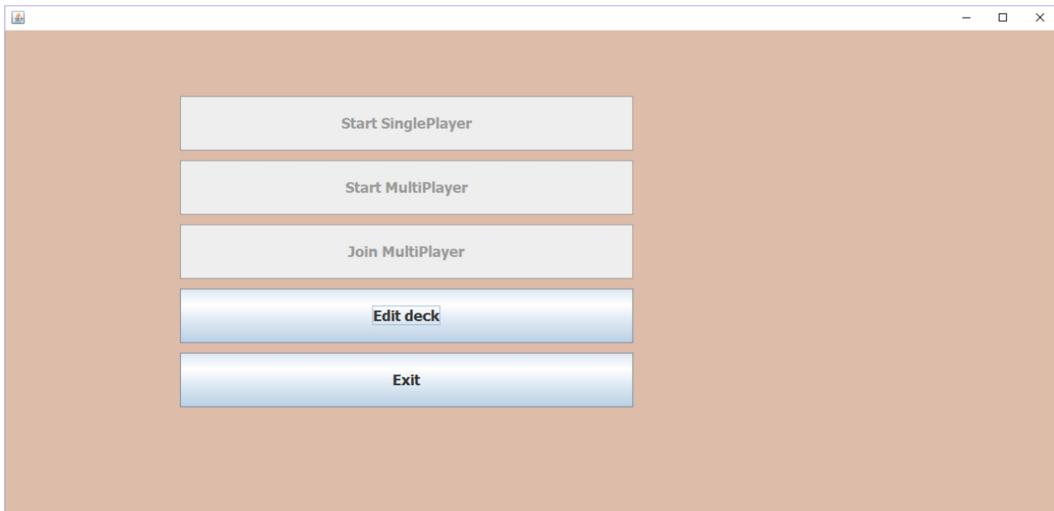


1. ábra Bejelentkezés

Itt a játékos megadhatja a nevét, ha már létezik ilyen felhasználó akkor a mentéstől folytatja, ha még nincs ilyen nevű játékos létrehozza.

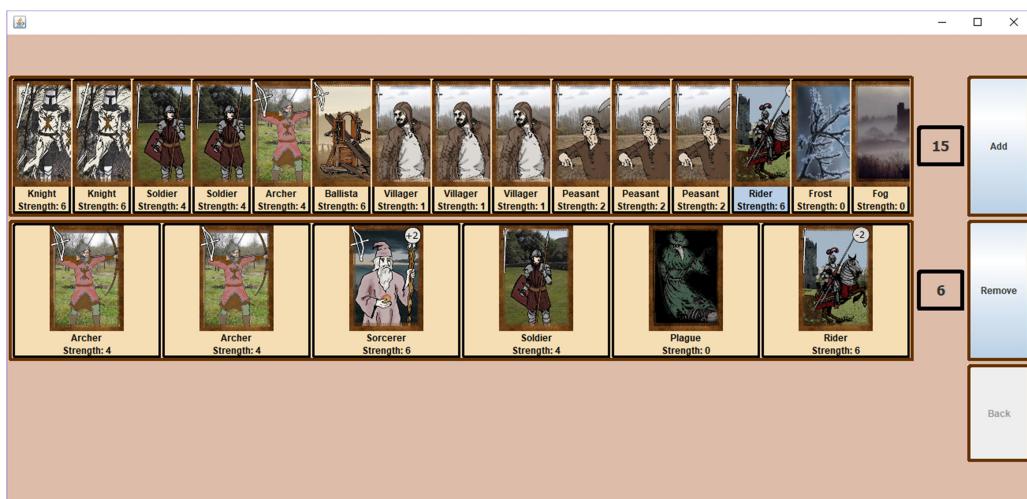
Pakli szerkesztése

Első bejelentkezés után a játék indítási gombok inaktívak, hogy elérhetőek legyenek legalább 20 lapot be kell rakknia a játékosnak a paklijába.



2. ábra Pakli szerkesztése 1.

Ezt az **Edit deck** nevű menüpontban teheti meg (2. ábra). A almenü (3. ábra) felső sorában a játékos azon kártyái találhatóak, amelyek még nincsenek hozzáadva a paklihoz. Az alsó sorban a pakli kártyái szerepelnek. Legalább 20 darab kártyát kell a paklihoz adni, ezt a kártyára való dupla kattintással, vagy a kártya kiválasztásával és utána az Add gomb megnyomásával teheti meg a felhasználó. A pakliból való kivétel az előbbihez hasonló módon történik, az Add gomb helyett a Remove gombbal vagy dupla kattintással.

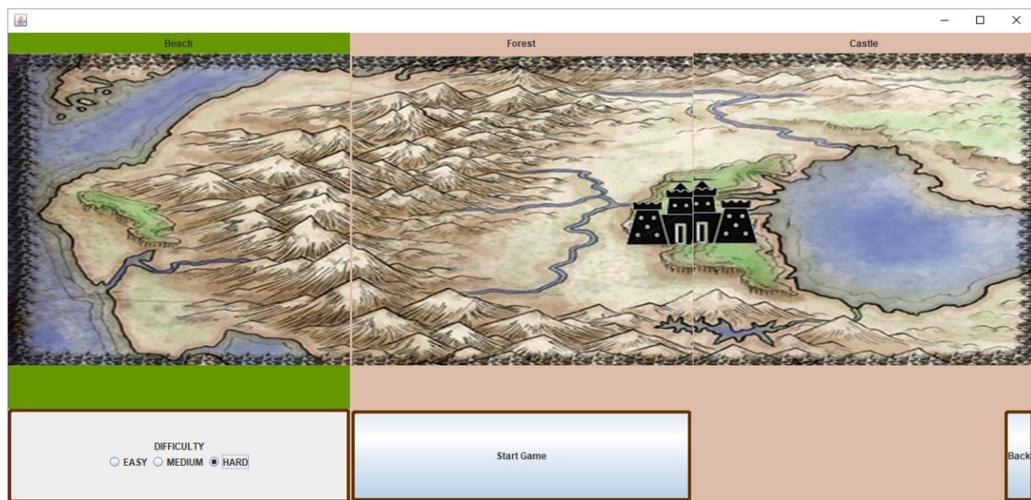


3. ábra Pakli szerkesztés 2.

A Back gombbal lehet visszalépni a főmenübe (2. ábra). A sorok jobb oldalán találhatóak a számlálók, amelyek azt mutatják hány darab kártya van az adott sorban.

Egyjátékos mód indítása

A pakli szerkesztése után elérhetővé válik a Start singleplayer gomb, ami a felhasználót a térképhez (4. ábra) viszi. Itt láthatja az egyjátékos módban melyik pálya következik, valamint kiválaszthatja a nehézségi szintet, ez lehet könnyű (Easy), közepes (Medium) és nehéz (Hard), majd elindíthatja a játékot a Start game gombbal.



4. ábra Egyjátékos mód indítása

A Back gomb a főmenübe viszi vissza a felhasználót.

Többjátékos mód indítása

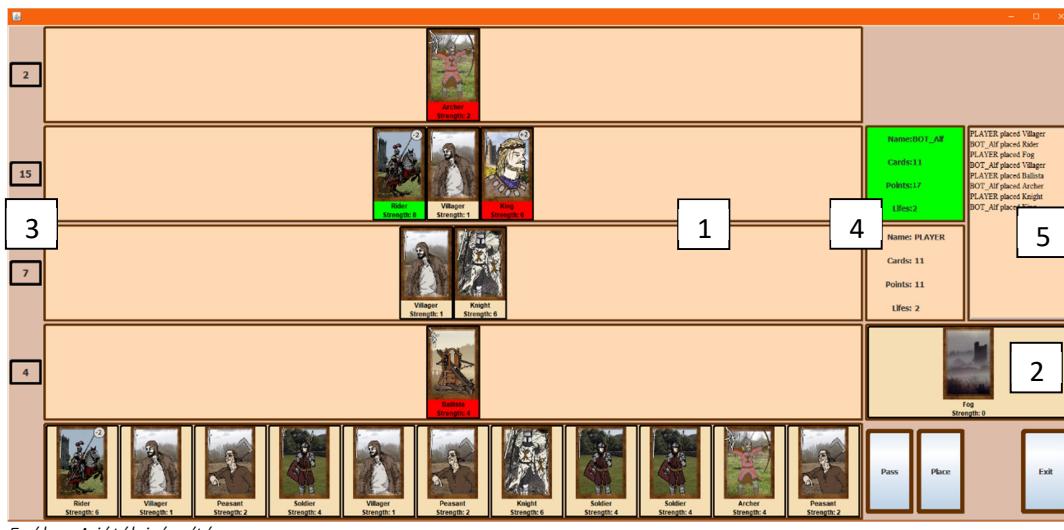
Többjátékos módot a Start MultiPlayer gombbal lehet indítani, ekkor megjelenik a játéktábla (5. ábra), ahol várni kell a másik játékos csatlakozására. Ha mindkét játékos csatlakozott elkezdődik a játék.

Többjátékos módhoz való csatlakozás

Csatlakozni akkor lehet egy játékhoz, ha az előtte létre lett hozva, ezt a Join MultiPlayer. Ilyenkor egyből a játék felületére kerül a felhasználó.

A játék irányítása

Amikor a játékoson van a sor, hogy rakjon lapot, elérhetővé válnak a Pass és a Place gombok. Egy kártyát kiválasztva a pakliból, a képernyő legalján lévő Place gombbal tehetjük le, illetve a kártyára való dupla kattintással. A Pass gombbal lehet passzolni, ebben az esetben a kör végéig nem rakhatunk le új lapot.



5. ábra A játék irányítása

A lerakott lapok a képernyő közepén található sorokban (5. ábra, 1), a speciális lapok a jobb oldalon található sorban (5. ábra, 2) jelennek meg.

A játék tábla bal oldalán találhatók a számlálók (5. ábra, 3), amelyek a sorok értékeit mutatják. Jobb oldalon található a két játékos panelje (5. ábra, 4), ezeken láthatóak a felhasználók nevei, maradék kártyáinak a száma, az összpontszámuk és az életeik száma. Amelyik játékos vezet annak a panelje zöld, döntetlen esetén mindenki barna és a keret pirosra vált, ha egy játékos passzol.

A játékos panelektől jobbra a napló (5. ábra, 5) található, ami a játékosok lépésein írja ki. Itt lehet visszakövetni melyik játékos milyen lapot rakott le, illetve ki nyerte a kört. Az Exit gombbal megszakíthatjuk a játékot.

2.4 Kártyák

Egy kártya felépítése

Egy kártya (6.ábra) áll egy képből, amelyen megtalálható melyik sorba tartozik, ezt a bal felső sarokban egy íj vagy kard jelzi, valamint látható milyen képessége van, ha van, ezt jobb felül egy szám jelzi. A kép alatt található a kártya neve és erőssége. Az erőssége változhat játék közben, ezt a szám változása mellett a háttér színe is jelzi, piros háttérrel, ha gyengébb, mint eredetileg és zölddel, ha erősebb.



6. ábra Kártya felépítése

| Közelharci egységek: | Távolharci egységek: | Speciális lapok: |
|---|--|---|
| Villager:  | Archer:  | Frost:  |
| Erő: 1 Képesség: nincs | Erő:4 Képesség: nincs | Képesség: minden közelharci sorára -2 pont minden lapra |
| Peasant:  | Ballista:  | Fog:  |
| Erő: 2 Képesség: nincs | Erő:6 Képesség: nincs | Képesség: minden távolharci sorára -2 pont minden lapra |
| Soldier:  | Sorcerer:  | Plague:  |
| Erő: 4 Képesség: nincs | Erő:6 Képesség: +2 | Képesség: törli az összes legyengébb lapot |

Knight:



Erő: 6 Képesség: nincs

Rider:



Erő: 6 Képesség: -2

Commander:



Erő: 8 Képesség: +1

King:



Erő: 8 Képesség: +2

3. Fejlesztői dokumentáció

3.1 A játék részletes leírása

A Trónfoglaló stratégiai kártyajáték, játszható egyjátékos, illetve többjátékos módban is, az utóbbi módban kettő emberi játékos játszhat egymás ellen. A játék két személyes, mindenki játékos egy általuk összerakott legalább 20 kártyából álló pakliból 15-15 véletlenszerű lapot húz fel. A kártyák közel- és távolharci egységek, ezt a bal felső részen egy kard vagy íj jelzi, időjárást változtató, illetve speciális képességű lapok lehetnek. Az egység lapoknak lehet képességük, amellyel egy sor erejét növelik vagy csökkentik, ezt a kártya jobb felső sarkában lévő szám jelzi. Az időjárást változtató lapok, a Kód és Fagy lapok, a távolharci és közelharci lapok erejét csökkentik mindenki játékosnál. A Járvány lap a lerakott kártyák közül törli a legkisebb értékű lapokat mindenki játékosnál. A fordulót az a játékos nyeri meg, akinek a lapjainak összértéke a nagyobb, a veszes elveszíti egy életét. Mindenki játékosnak kettő élete van, akinek előbb elfogy az élete, veszít, így a játék 2 vagy 3 fordulós. Döntetlen játék esetén az egyjátékos módban újra játszás következik.

A játékos a pakliját az Edit deck menüben tudja alakítani, a jobb oldali számlálók mutatják hány darab kártya van a paklin kívül, illetve a pakliban. Amíg nincs a pakliban legalább 20 lap nem lehet játékot indítani.

Egyjátékos módban a felhasználó a számítógép ellen játszhat, egy választott nehézségi szinten, ami lehet könnyű, közepes vagy nehéz. A szintek között a különbséget a gép által használt taktikák jelentik, könnyű nehézségi szinten a legegyszerűbbeket használja, nehéz szinten pedig legösszetettebb taktikákat. Könnyű szinten a gépi játékos csak egy egyszerű algoritmus alapján választja ki a legkisebb értékű lapot, amit lerakva még ó vezet. Közepes nehézségen egy véletlenszerű szám alapján van esélye, hogy csak a legkisebb lapjait rakja le, így kicsalva az ellenség nagyobb lapjait és előnyhöz jutva. Nehéz szinten a AI statisztikákat felhasználva számol esélyt a speciális kártyáinak a lerakására, ezt a statisztikát a felhasználó játékából készíti a program. Ha a játékos felhasznál egy speciális kártyát (fagy, kód, járvány), akkor a statisztikába bekerül mekkora előnyhöz jutott vele a játékos.

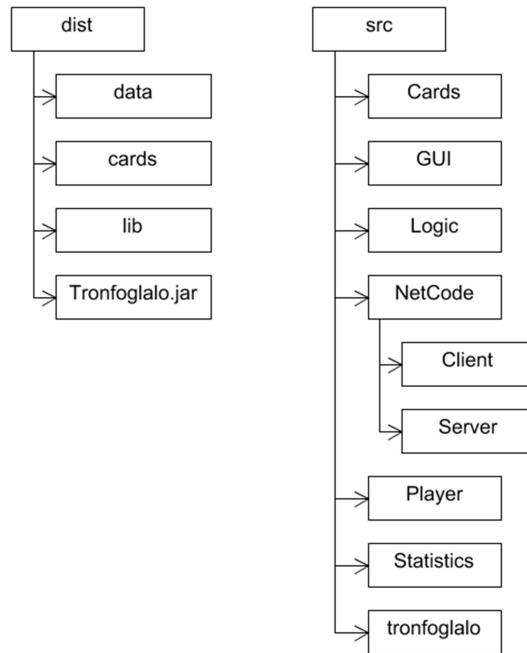
A játék célja egy térkép összes területének elfoglalása. A térkép megnyitása után kijelzi a játék, hogy melyik pálya következik, és ezt lehet elindítani. minden elfoglalt terület után a játékos új lapokat kap, melyet elhelyezhet a paklijába. A paklit a főmenüből elérhető Edit Deck almenüben szerkesztheti a játékos. minden vereség után egy véletlenszerű lapot elveszít. Ezt először a fel nem használt lapjai közül teszi meg a program, majd, ha azok elfogytak, a paklijából veszi el a lapokat.

Ha a játékosnak nem marad elegendő lapja, akkor veszít, ebben az esetben a lapjai az alap paklira cserélődnek és a mentése törlődik. minden lejátszott játék után a jelenlegi állás mentésre kerül és a legközelebbi indításnál innen folytathatja a játékot. minden új felhasználó az elejéről kezdi a játékot. Egy játék után a játékos visszakerül a főmenübe, ahonnan szerkesztheti a pakliját, illetve folytathatja a játékot. Az itt megszerzett lapokat többjátékos módban is használhatják.

Többjátékos módban két ember játszhat egymás ellen. Az egyik játékosnak el kell indítani egy szervert a Start Multiplayer gombra kattintva, utána a másik játékos csatlakozhat a Join Multiplayer gomb megnyomásával. Ebben a módban nincsen térkép, a két játékos egy partit játszik le. A nyertes nem kap új lapot és a vesztes nem veszti el egyik lapját sem. Döntetlen játék esetén nincsen újrajátszás.

3.2 A mappaszerkezet felépítése

A játék mappaszerkezete (7. ábra) egyszerű, a data mappában vannak az adatbázisok és a térképhez tartozó képek, a cards mappában a kártyák képei találhatóak.



7. ábra Mappaszerkezet

A forráskód mappaszerkezete a következő:

Cards: a kártyákkal kapcsolatos osztályokat tartalmazza.

GUI: a grafikus felület osztályait tartalmazza.

Logic: a controller és logika elemeit tartalmazza.

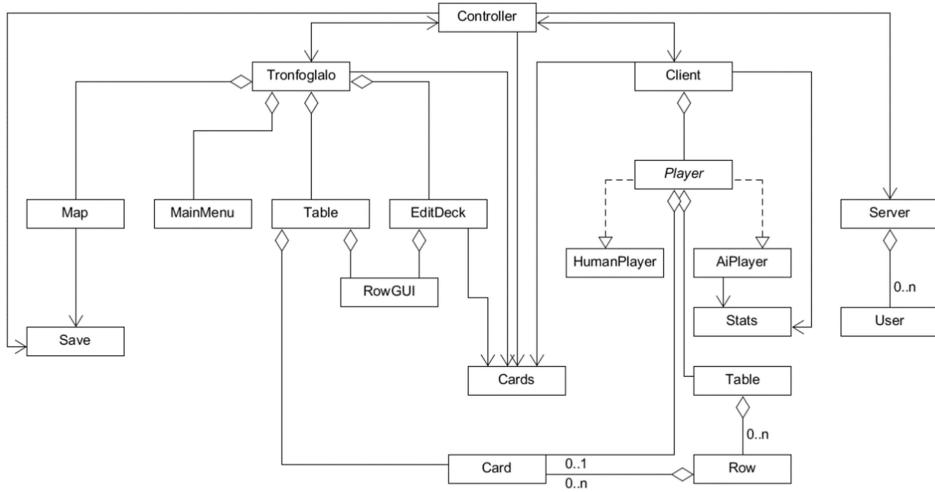
NetCode: a Client és Server található benne, ezek almappákban vannak tárolva.

Player: az emberi és gépi játékos elemeit tartalmazza.

Statistics: a statisztika elemeit tartalmazza.

tronfoglalo: az indító és adatbázis építő osztályokat tartalmazza.

3.3 A program szerkezete



9. ábra A program szerkezete

A program vezérléséért a Controller nevű statikus osztály a felelős. Ennek a metódusai irányítják a Tronfoglalo és a Client osztályokat, ezeket indításkor kapja meg a Controller osztály.

A Client osztály kommunikál a szerverrel, ez ütemezi a játék eseményeit. A Controller osztályon keresztül frissíti a grafikus felületet.

A Client-ben található egy Player típusú változó, ezt a példányosításakor hozza létre, lehet HumanPlayer vagy AiPlayer a játékos típusától függően.

A Player osztályban tárolva vannak a játékos kártyái és a játéktábla, ebből frissíti a grafikus felületet a játék.

A Table osztály Row osztályokból áll, ezek a játéktábla sorait és azoknak az adatait tárolják.

Az AiPlayer eléri a Stats osztályon keresztül a statistics adatbázist, amelyből a kártyaválasztáshoz használ fel adatokat. Ezeket az adatokat a Client osztály frissíti.

A Client a Cards osztályból éri el a kártyákat tartalmazó adatbázisokat, ezek között van a játékos kártyáinak a tárolására használt adatbázisok is.

A Cards egy statikus osztály, amely a kártyák tárolásáért felel. Három adatbázisban tárolja őket, cards, ebben vannak a kártyák adatai, deck, ebben vannak a játékos paklijában lévő kártyák, myCards, amiben a játékos maradék kártyája található.

A Tronfoglalo osztály a grafikus felület vezérléséért felelős, ennek a részei a Map, MainMenu, Table, EditDeck.

A Map osztály az egyjátékos mód térképéért és a mentésért felelős, az utóbbit a Save osztály kezeli.

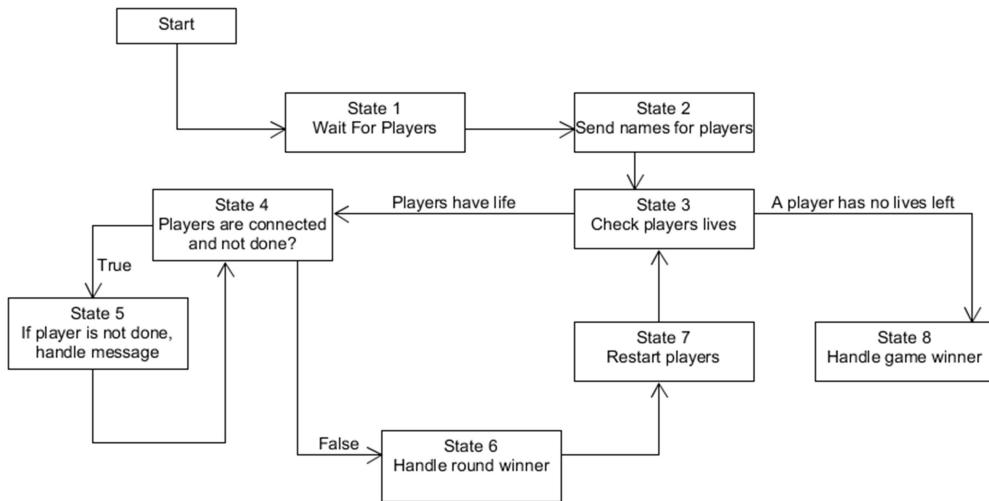
A MainMenu a főmenüt tartalmazza, innen indítható a játék és szerkeszthető a pakli.

A Table osztály maga a játéktábla, ebben a sorokat RowGUI osztályok alkotják.

Az EditDeck osztályban szintén RowGUI osztályok alkotják a sorokat. Itt módosíthatjuk a paklit.

3.4 Szerver – Kliens

A szerver - kliens működést 3 állomány alkotja: Client (13. ábra), Server (12. ábra), User (11. ábra). A szerver a jelenlegi működés alapján, egy a kódban megadott porton indul el lokális hálózaton. A szerver - kliens rész a JAVA alap socket [3] kommunikációjával van megvalósítva.



10. ábra A szerver UML diagramja

Szerver:

A szerver létrehozható megadott porttal vagy véletlenszerű szabad porttal is. Ez a játékban csak az előre megadott porttal van használva. Egy létrehozott és elindított szerver addig fut amíg mindenktől játékosnak van élete. Egy kört addig futtat amíg mindenktől játékos nem passzol. A két játékos között váltogatva fogad üzeneteket. Ha az egyik játékos passzol csak a másiktól fogad új üzenetet. A bejövő üzeneteket feldolgozza majd a játékosokat értesíti a változásokról.

A szerver indításakor várakozik a két játékos csatlakozására, ez a `waitForPlayers` metódus. Amíg

| Netcode.Server::User |
|------------------------------|
| -playerNo = 0: int |
| -name: String |
| -s: Socket |
| -sc: Scanner |
| -pw: PrintWriter |
| -notDone = true: boolean |
| -LOG = false: boolean |
| +User(ServerSocket ss): ctor |
| -LOG(String log): void |
| +getName(): String |
| +getCards(): String |
| +getLifes(): int |
| +send(String msg): void |
| +receive(): String |
| -notDone(): boolean |
| -isConnected(): boolean |
| -isDone(): void |
| -getPoints(): int |
| -removeLife(String p): void |
| -restart(): void |
| -disconnected(): void |

11. ábra Netcode.Server.User

a játékosok száma nem éri el a kettőt, új játékost fogad, egy játékost a User osztály valósít meg. Ennek a létrehozása elindítja a várakozást új játékosra. Csatlakozáskor a játékos elküldi a nevét is a szervernek. Miután a két játékos csatlakozott a szerver elküldi nekik a neveket.

| Netcode.Server::Server | |
|---|--|
| -PORT = 0: int | |
| -ss: ServerSocket | |
| -players = new ArrayList<User>(): ArrayList<User> | |
| -LOG = false: boolean | |
| +Server(int port): ctor | |
| +Server(): ctor | |
| -LOG(String log): void | |
| -waitForPlayers(): void | |
| -sendPlayersWait(): boolean | |
| -sendPlayersEnded(): void | |
| -sendPlayerGo(int p): void | |
| -receivePlayer(int p): String | |
| -sendPlayersCard(String card, int from): void | |
| -sendPlayersLives(): void | |
| -sendPlayersName(): void | |
| -sendPlayersCardsNo(): void | |
| +run(): void | |

12. ábra Netcode.Server.Server

Amíg minden játékosnak van élete, elküldi a felhasználóknak ez ellenfél életeinek számát. Majd, ha a játékosok csatlakozva vannak és legalább az egyikük nem passzol, fogadja az üzenetet és feldolgozza. Ha az egyik játékos bontja a kapcsolatot, kilép a ciklusból és befejezi a játékot. A végén bezárja a szerver socketet.

Kliens:

A kliens futtatásakor első lépésként csatlakozik a szerverhez, majd elkezdi az üzenetek fogadását.

Csatlakozásnál, ha sikeres a kapcsolat létrehozása, elküldi a szervernek a saját nevét, ezután vár az szerverre, hogy elküldje neki az ellenség nevét.

Az üzenet fogadását új szalon indítja a program, itt vár, hogy a szerver elküldje a következő üzenetet. Az üzenetet megkapva az feldolgozza.

Az üzenet lehet:

- WAIT: ennek hatására frissíti a GUI-t, ha a kliens nem AI, válaszul pedig egy READY üzenetet küld
- GO: ennek hatására a GUI engedélyezi a kártyák gombjait, illetve AI esetén elkéri tőle a lerakni kívánt lapot
- ENDED: ha nem AI a játékos kiírja a győztest
- GETCARDS: elküldi a szervernek a játékos kezében lévő lapok számát

- SETCARDS: fogadja a szervertől hány lapja van az ellenségnak
- GETLIVES: elküldi a szervernek a játékos életeinek számát
- SETLIVE: fogadja a szervertől hány élete van az ellenségnak
- GETPOINTS: elküldi a szervernek a játékos pontjait
- REMOVELIFE: törli az következő üzenetben érkező játékos egy életét
- RESET: visszaállítja a táblát a kezdőállapotra
- default: ha egyik feljebb található üzenetre sem illik, akkor felbontja. Az üzenet első karaktere a küldő száma, ez lehet 1 vagy 2, a maradék pedig egy kártya azonosítója vagy a DONE karakterlánc, amely esetben az ellenfél passzolt.

| Netcode.Client::Client |
|---|
| <pre>-s: Socket -sc: Scanner -pw: PrintWriter -name: String -address: String -PORT: int -player: Player.Player -msg = "": String -AI = false: boolean +Client(String address, int PORT, String name, String type, List<Card> deck): ctor +getName(): String +connect(): void +receiveMsg(): void +getRow(int i): Row +getHand(): List<Card> +addToTable(Card c, int p): void +getLifes(): int +sendCard(Card card): void +run(): void -received(String msg): void +reset(): void +getMyPoints(): int +getEnemyPoints(): int +getDeck(): List<Card> +sendExit(): void</pre> |

13. ábra Netcode.Client.Client

A kapott üzenet alapján a kliens meghívja a Controller megfelelő függvényét a feladat végrehajtásához.

3.5 A játék logikája

A játék logikájáért elsősorban a Logic mappában található Controller (14. ábra), Row és Table a felelős, a játék ütemezéséért a Client és Server.

A Controller-nek összekötő szerepe van a Tronfoglalo és a Client között, minden kettő referenciáját tárolja az osztály. Itt tároljuk a deck változóban a gépi játékoshoz tartozó paklikat, ebből minden sor egy pályához ad meg paklit, a prize változóban található a pályák teljesítéséért járó plusz kártyák. Ugyanitt tároljuk a játékos pakliját és többi kártyáját is, ezek a deck és myCards változók, ezeket az adatbázisból frissítjük. A különböző játékmódok indítója is itt található.

| Logic::Controller | |
|--|--|
| -tronfoglalo: GUI.Tronfoglalo | |
| -client: Client | |
| -deck = new ArrayList<Card>(): List<Card> | |
| -myCards = new ArrayList<Card>(): List<Card> | |
| -enemyPassed = false: boolean | |
| -difficulty = 3: int | |
| -map = -1: int | |
| -prize = { { 4, 6 }, { 3, 1 }, { 7, 3 } }: int[][], | |
| -name: String | |
| -decks = { { 5, 5, 5, 2, 2, 2, 1, 1, 3, 4, 7, 8, 8, 9, 9, 10, 10, 11, 12, 13 }, //{1,1,2,2,3,4,5,5,6,6,7,9,9,11,12,13}, | |
| { 1, 1, 1, 2, 2, 2, 3, 3, 5, 5, 9, 9, 9, 10, 10, 10, 11, 12, 13 }, { 1, 1, 1, 2, 2, 2, 3, 3, 4, 5, 5, 6, 6, 6, 7, 9, 9, 11, 12, 13 } }: int[][], | |
| +addGUI(GUI.Tronfoglalo tronfoglalo, List<Card> cards, List<Card> deck, String name): void | |
| +addClient(Client client): void | |
| +refresh(): void | |
| +refreshRow(int r): void | |
| +refreshHandRow(): void | |
| +sendCard(Card card): void | |
| +getHand(): List<Card> | |
| +getRow(int i): Row | |
| +removeCard(Card card): void | |
| +disableHand(): void | |
| +enableHand(): void | |
| +add.ToTable(Card c, int p): void | |
| +setEnemyName(String name): void | |
| +getEnemyName(): String | |
| +setEnemyCards(String cards): void | |
| +setEnemyLifes(String lifes): void | |
| +setMyCards(): void | |
| +setMyLifes(): void | |
| +reset(): void | |
| +setPoints(): void | |
| +startServer(int PORT): void | |
| +openMap(): void | |
| +closeMap(): void | |
| +startSinglePlayer(): void | |
| +startMultiPlayer(): void | |
| +joinMultiPlayer(): void | |
| +editDeck(): void | |
| +editDeckBack(): void | |
| +saveDeck(List<Card> cards): void | |
| +getDeck(): List<Card> | |
| +getCards(): List<Card> | |
| +addToDeck(Card selected): void | |
| +removeFromDeck(Card selected): void | |
| +showWinner(int playerOneLives, int playerTwoLives): void | |
| +passed(int from): void | |
| +getEnemyPassed(): boolean | |
| +log(String text): void | |
| +getDifficulty(): int | |
| +setDifficulty(int d): void | |
| +exitGame(): void | |
| +exit(): void | |
| +getName(): String | |

14. ábra Controller

A Table (15. ábra) osztály Row osztályokból áll össze. Ebben tároljuk a játéktábla sorait, ebből 4 sor az egység lapok helye, 1 pedig a különleges lapoké. Innen lehet lekérni a sorok értékét, valamint a játékosok összpontszámát. A kártyák hozzáadása is itt történik, ha van képessége egy kártyának akkor a képessége végrehajtódik, ha nincs akkor a kártyát berakjuk a neki megfelelő sorba és

| Logic::Table |
|---------------------------------------|
| -rows = new Row[5]: Row[] |
| +Table(): ctor |
| +Table(Table that): ctor |
| +getPlayerOnePoints(): int |
| +getPlayerTwoPoints(): int |
| +tryCard(Card card, int p): int |
| +addCard(Card card, int player): void |
| +removeCard(Card card): void |
| +getRow(int r): Row |
| +reset(): void |
| -deleteWeakest(): void |
| -applyFrost(): void |
| -applyFog(): void |

15. ábra Logic.Table

frissítjük a sor pontszámát. Kártyák törlése is itt történik, ez a járvány laphoz szükséges.

A Table továbbá tartalmazza a gépi játékos által használt függvényeket is, például a tryCard függvényt, ez visszaadja egy kártyát lerakva milyen előnyhöz jutna a játékos. Ehhez a táblát lemásolja és abba dolgozik.

A Row (16. ábra) ami a Table-t alkotja, a tábla egy sorát valósítja meg. Ez a sorban található kártyákat tárolja, van egy erő állapot mutatója, powerState és egy integer változő, amiben a pontszám van tárolva. Lehet kártyát hozzáadni és törölni, minden esetben változhat az erő állapot, így,

ha a kártya plusz ereje nem nulla, akkor ezt frissíti, majd a pontokat is frissíti. El lehet kérni a benne lévő kártyákat, és vissza lehet állítani a kezdő állapotára, ami töri a kártyákat és nullázza az erő állapotot és pontokat.

A logikához tartozik még az AiPlayer (17. ábra) is, ez a gépi játékost valósítja meg. A fő metódusa a getCard ami visszaadja a gép által választott kártyát, ehhez a getValue függvényt is felhasználja, ami egy kártya jelenlegi értékét adja vissza.

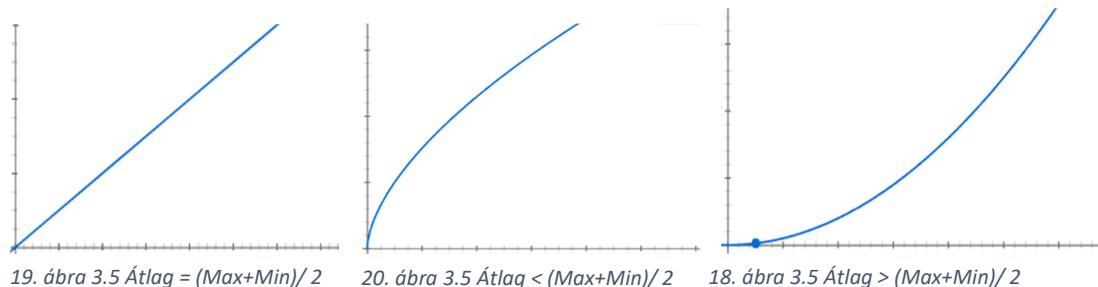
| Logic::Row |
|--|
| -points: int |
| -cards = new ArrayList<Card>(): List<Card> |
| -powerState: int |
| +Row(): ctor |
| +Row(Row that): ctor |
| +addCard(Card card): void |
| +removeCard(Card card): void |
| +powerChange(int change): void |
| +getCards(): List<Card> |
| -pointsUpdate(): void |
| +getPoints(): int |
| +reset(): void |

16. ábra Logic.Row

A nehézségi szintek egyes választási elágazásokat kapcsol be és ki. A legegyszerűbb fokozaton a gép csak a legkisebb értékű lapot választja ki, amivel még éppen vezetésre állna. A lapjait ezért növekvő sorrendben tárolja, ezzel csökkentve a műveletiigényét. A lapokat a kártya kapásakor rendezi. Közepes nehézségen 50% az esélye, hogy kis lapokat kezd el kijátszani a gép, ezzel kicsalva a játékostól, hogy nagyobb lapokat rakjon. Ezt a taktikát csak abban az esetben használja, ha több

mint egy élete van. Ezt csak egy pár lap lerakásáig használja utána passzol. A legnehezebb fokozaton használja statisztikát a speciális lapjainak a lerakásához. Ebben egy exponenciális függvény adja vissza az eddigi statisztikák szerint mekkora az esély, hogy lerakja a lapot. Ezt az eddigi maximum, átlag és a minimum szerzett előny szerint számolja ki az emberi játékos akciói alapján. Amikor egy gépi kliens kap egy speciális lapot a játékostól, akkor a játékos által szerzett előnnyel frissíti az adatbázist. Ezért, ha a játékos kis előnyért is felhasználja speciális lapjait a gép is fel fogja, ha csak nagy előnyért a gép is csak nagy előnyért fogja. A függvény így háromféle képen nézhet ki, az átlag előny megegyezik (19.ábra), az átlag kisebb (21. ábra)

vagy az átlag nagyobb (18. ábra) a maximum és minimum összegének a felével



A `getCard` meghívásakor egy másodpercet vár a program, hogy játék átláthatóbb legyen.

| Player::AiPlayer |
|--|
| <pre>-playSmallCards = false: boolean -startHandSize: int -placedCards = 0: int -handTable: Logic.Table -unitCard = 11: int +AiPlayer(String name, List<Card> deck): ctor +addCard(Card c): void +getCard(): Card +getHand(): List<Card> +getRow(int r): Row +addToTable(Card card, int player): void -getValue(Card card): int</pre> |

17. ábra Player.AiPlayer

3.6 Adatbázisok és felépítésük

A játékhoz öt darab adatbázis tartozik: cards,deck,mycards,statistics és save. A cards, deck, mycards kezeléséért a Cards osztály felel.

Cards:

A cards (21. ábra) adatbázis a kártyák adatait tárolja, ezt az init függvény meghívásával tölti fel a rendszer, ezt elég ez első futtatás esetén elvégezni, illetve ha a kártyákon változtatást hajtunk végre.

| Cards::Cards |
|---|
| -cards = new ArrayList<Card>(): List<Card> |
| -url = "jdbc:sqlite:data/cards.db": String |
| -userName = "root": String |
| -password = "root": String |
| +createDatabase(): void |
| -insertNewCard(String name, int strength, String picture, int power, int row, String ability): void |
| -insertNewCard(String name, int strength, String picture, int power, int row): void |
| +insertIntoMyCards(String name, int cardID): void |
| +insertIntoDeck(String name, int cardID): void |
| +moveToDeck(String name, int id): void |
| +deleteFromDeck(String name, int id): void |
| +resetPlayer(String name): void |
| +moveToMycards(String name, int id): void |
| +deleteFromMycards(String name, int id): void |
| +getCard(int ID): Card |
| +initPlayer(String name): void |
| +init(): void |
| +getCards(String playername, String table): List<Card> |

21. ábra Cards.Cards

A cards adattagjai:

- cardID: integer → a kártya azonosítója
- name: string → a kártya neve
- strength: integer → a kártya alap ereje
- picture: string → a kártya képének a neve
- power: integer → a kártya erejének képessége
- row: integer → a kártya sorát adja meg, 0 közelharci, 1 távolharci
- ability: string → nem kötelező, a speciális képességet adja meg

Deck:

A játékos paklijának kártyáit tárolja.

A deck adattagjai:

- `playerName: string` → a játékos neve, akihez tartozik
- `id: integer` → azonosító
- `cardID: integer` → a kártya azonosítója

MyCards:

A játékos pakliján kívüli kártyáit tárolja.

A myCards adattagjai:

- `playerName: string` → a játékos neve, akihez tartozik
- `id: integer` → azonosító
- `cardID: integer` → a kártya azonosítója

A `statistics` (22. ábra) adatbázis a `Stats` osztályon keresztül érhető el. Ez a speciális kártyákkal kapcsolatos statisztikákat tartalmaz, az `AIPlayer` ennek a segítségével választ a kártyák közül, a nehéz játékmódban.

| Statistics::Stats |
|--|
| <code>-stats = new ArrayList<Stat>(): List<Stat></code> |
| <code>-url = "jdbc:sqlite:data/stats.db": String</code> |
| <code>-userName = "root": String</code> |
| <code>-password = "root": String</code> |
| <code>+createDatabase(): void</code> |
| <code>+init(): void</code> |
| <code>-insertNewStat(String name, double avg, int max, int min, int smpls, double mult, double exp): void</code> |
| <code>+getStat(String name): Stat</code> |
| <code>-refreshStat(Stat s): void</code> |
| |

22. ábra `Statistics.Stats`

A statistics adattagai:

- name: string → a kártya neve
- average: float → az átlag milyen plusz előnyre tett szert vele a játékos
- max: integer → a vele szerzett eddigi legnagyobb előny
- min: integer → a vele szerzett eddigi legkisebb előny
- samples: integer → eddig hány mintát tárolunk
- mult: float → az adatokra illesztett függvény szorzója
- exp: float → az adatokra illesztett függvény paramétere

Egy statisztikát a kártyánév alapján kérhetünk le, ez egy Stat (23.ábra) típusú elemet ad vissza.

| Statistics::Stat |
|---|
| <pre>-name: String -avg: double -min: int -max: int -samples: int -exp: double -mult: double +Stat(String name, double avg, int max, int min, int smpls, double mult, double exp): ctor +getChance(int points): double +calcFunction(): void +addStat(int p): void -Function(double m, double e, double p): double ~getName(): String +getAvg(): double +getMax(): int +getMin(): int ~getSmppls(): int ~getMult(): double ~getExp(): double</pre> |

23. ábra Statistics.Stat

A saves (24. ábra) adatbázis az egyjátékos játékmód mentéseit kezeli.

| Logic::Save |
|--|
| -stats = new ArrayList<Stat>(): List<Stat> |
| -url = "jdbc:sqlite:data/saves.db": String |
| -userName = "root": String |
| -password = "root": String |
| -name: String |
| +createDatabase(): void |
| +init(): void |
| +reset(String name): void |
| +refreshSave(String name, int map): void |
| +addNewSave(String name): void |
| +getSave(String name): int |

24. ábra 3.6 Logic.Save

A saves adattagai:

- name: string → a felhasználó neve
- map: integer → a felhasználóhoz tartozóan melyik pályán tart

3.7 Grafikus felhasználói felület

A grafikus felhasználói felület 6 osztályból áll össze:

- Tronfoglalo (25. ábra): ez ad keretet az egész játéknak, ez indul el a játékkal. Létrehozásakor bekéri a játékos nevét, majd eszerint folytatja a további lépéseket. Lekéri a mycards és deck adatbázisuktól a kapott névhez tartozó kártyákat, majd hozzáadja saját magát a statikus Controller osztályhoz. A többi osztály a RowGUI kivételével itt kerül példányosításra.

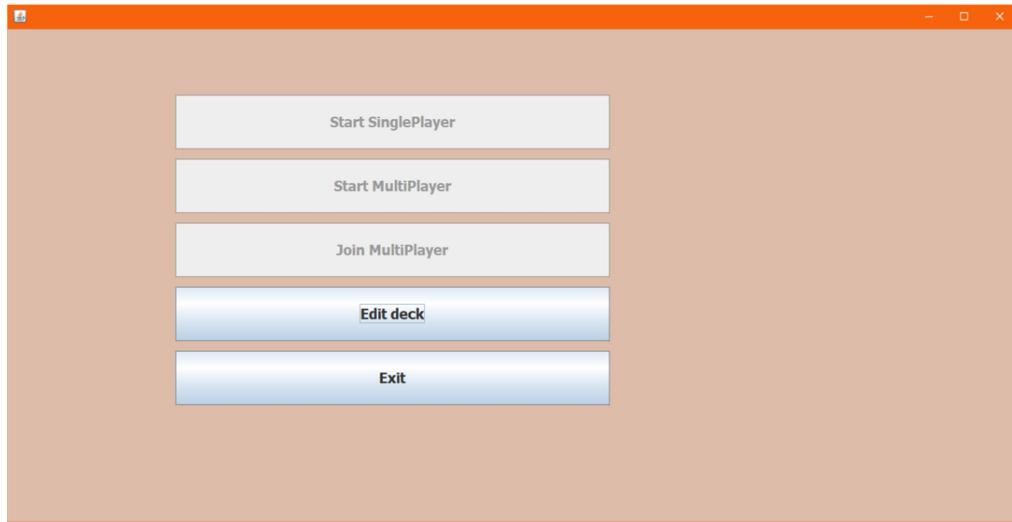
| GUI::Tronfoglalo | |
|--|------|
| -started = false: boolean | |
| -name: String | |
| -editDeck1: GUI.EditDeck | |
| -mainMenu1: GUI.MainMenu | |
| -map1: GUI.Map | |
| -table1: GUI.Table | |
| +Tronfoglalo(): ctor | |
| +openMap(): void | |
| +closeMap(): void | |
| +startGame(String mode, List<Card> deck): void | |
| +startAI(List<Card> deck): void | |
| +joinMultiPlayer(int PORT, String IP, List<Card> deck): void | |
| -initComponents(): // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents | void |
| -formComponentResized(java.awt.event.ComponentEvent evt): void | |
| +setMyName(String name): void | |
| +setEnemyName(String name): void | |
| +setEnemyCards(String cards): void | |
| +refreshHandRow(): void | |
| +getTable(): Table | |
| +refreshRow(List<Card> cards, int row, int points): void | |
| +run(): void | |
| +disableHand(): void | |
| +enableHand(): void | |
| +setEnemyLives(String lifes): void | |
| +setMyCards(String size): void | |
| +setMyLifes(String string): void | |
| +setPoints(int myPoints, int enemyPoints): void | |
| +editDeck(): void | |
| +editDeckBack(): void | |
| +enemyPassed(): void | |
| +showWinner(int playerOneLives, int playerTwoLives): void | |
| +resetEnemyPassed(): void | |
| +resetPassed(): void | |
| +log(String text): void | |
| +getEnemyName(): String | |
| +gameExit(): void | |
| +exit(): void | |
| +showGameLost(): void | |
| +cardNumberCheck(): void | |

25. ábra GUI.Tronfoglalo

- MainMenu (26. ábra): a főmenüt valósítja meg, itt választhat a játékos milyen játékmódot akar indítani, vagy hogy szerkesztené a pakliját. Ha a játékos paklijában nincs legalább 20 darab lap a Start Singleplayer, Start Multiplayer és Join Multiplayer gombok nem elérhetőek.

| GUI::MainMenu |
|---|
| -editDeck: javax.swing.JButton |
| -exit: javax.swing.JButton |
| -joinMultiPlayer: javax.swing.JButton |
| -startMultiPlayer: javax.swing.JButton |
| -startSinglePlayer: javax.swing.JButton |
| +MainMenu(): ctor |
| -initComponents(): void |
| -startSinglePlayerActionPerformed(java.awt.event.ActionEvent evt): void |
| -startMultiPlayerActionPerformed(java.awt.event.ActionEvent evt): void |
| -joinMultiPlayerActionPerformed(java.awt.event.ActionEvent evt): void |
| -editDeckActionPerformed(java.awt.event.ActionEvent evt): void |
| -exitActionPerformed(java.awt.event.ActionEvent evt): void |
| +cardNumberCheck(): void |

26. ábra GUI.MainMenu



27. ábra A főmenü képe

- EditDeck (26. ábra): itt tudja a felhasználó szerkeszteni a pakliját, ez két RowGUI-ból és gombokból áll. Ha a pakliban nincs elegendő lap, nem engedi a felhasználót vissza a fómenübe, ehhez a számlálók a soroktól jobbra található.

| GUI::EditDeck |
|--|
| <pre> -inMenu = false: boolean -addButton: javax.swing.JButton -backButton: javax.swing.JButton -cardsNO: javax.swing.JLabel -cardsRow: GUI.RowGui -deckNO: javax.swing.JLabel -deckRow: GUI.RowGui -removeButton: javax.swing.JButton +EditDeck(): ctor +init(): void -initComponents(): void -formComponentResized(java.awt.event.ComponentEvent evt): void -addButtonActionPerformed(java.awt.event.ActionEvent evt): void -backButtonActionPerformed(java.awt.event.ActionEvent evt): void -removeButtonActionPerformed(java.awt.event.ActionEvent evt): void -resized(): void </pre> |

28. ábra GUI.EditDeck

- Map: az egyjátékos móhoz tartozó térképet jeleníti meg. Egy nehézségi szint választó panel, egy indító gomb, egy visszagomb és a térkép található rajta. Amelyik pálya következik annak a háttere zöldre vált, ezt a pályát tudjuk elindítani.

| GUI::Map |
|--|
| <pre> ~p1 = new JLabel(): JLabel ~p2 = new JLabel(): JLabel ~p3 = new JLabel(): JLabel -backButton: javax.swing.JButton -diff1: javax.swing.JRadioButton -diff2: javax.swing.JRadioButton -diff3: javax.swing.JRadioButton -difficultyButtons: javax.swing.ButtonGroup -difficultyLabel: javax.swing.JLabel -jPanel1: javax.swing.JPanel -sector1: javax.swing.JPanel -sector2: javax.swing.JPanel -sector3: javax.swing.JPanel -startButton: javax.swing.JButton +Map(): ctor +display(): void -getScaledImage(Image srclmg, int w, int h): Image -initComponents(): void -formComponentResized(java.awt.event.ComponentEvent evt): void -backButtonActionPerformed(java.awt.event.ActionEvent evt): void -startButtonActionPerformed(java.awt.event.ActionEvent evt): void </pre> |

29. ábra GUI.Map



30. ábra A tértkép

- Table (31 ábra): ez az osztály valósítja meg magát a játékmezőt. Gombokból, RowGUI-kból, szövegmezőből és feliratokból áll. Bal oldalon

találhatóak a számlálók, amik az egyes sorok értékét jelölik, tőlük jobbra a játéktábla, ez négy sorból áll, kettő közelharci és kettő távolharci sorból. Jobbra mellette a játékosok paneljei, illetve a speciális kártyák sora található. Jobb oldalon található a játéknapló, ez mutatja a játékosok előző lépéseit.

| GUI::Table | |
|--|--|
| -p = 0: int +enabled = false: boolean -sb = new StringBuilder(): StringBuilder -PlayerCards: javax.swing.JLabel -PlayerCards1: javax.swing.JLabel -enemyBackPoint: javax.swing.JLabel -enemyBackRow: GUI.RowGui -enemyCards: javax.swing.JLabel -enemyFrontPoint: javax.swing.JLabel -enemyFrontRow: GUI.RowGui -enemyLifes: javax.swing.JLabel -enemyName: javax.swing.JLabel -enemyPoints: javax.swing.JLabel -exitButton: javax.swing.JButton -handRow: GUI.RowGui -logPane: javax.swing.JPanel -logScrollPane: javax.swing.JScrollPane -logTextPane: javax.swing.JTextPane -myBackPoint: javax.swing.JLabel -myBackRow: GUI.RowGui -myCards: javax.swing.JLabel -myFrontPoint: javax.swing.JLabel -myFrontRow: GUI.RowGui -myLifes: javax.swing.JLabel -myName: javax.swing.JLabel -myPoints: javax.swing.JLabel -passButton: javax.swing.JButton -placeButton: javax.swing.JButton -playerLifes: javax.swing.JLabel -playerLifes1: javax.swing.JLabel -playerName: javax.swing.JLabel -playerName1: javax.swing.JLabel -playerOnePanel: javax.swing.JPanel -playerPoints: javax.swing.JLabel -playerPoints1: javax.swing.JLabel -playerTwoPanel: javax.swing.JPanel -weatherRow: GUI.RowGui | |
| +Table(): ctor +setMyName(String name): void +getMyName(): String +getEnemyName(): String -initComponents(): void -exitButtonActionPerformed(java.awt.event.ActionEvent evt): void -passButtonActionPerformed (java.awt.event.ActionEvent evt): void -formComponentResized(java.awt.event.ComponentEvent evt): void -placeButtonActionPerformed(java.awt.event.ActionEvent evt): void +refreshHandRow(): void +refreshRow(List<Card> cards, int row, int points): void +enableHand(): void +disableHand(): void +setEnemyName(String name): void -setEnemyCards(String cards): void -setEnemyLifes(String lifes): void -setMyCards(String size): void -setMyLifes(String string): void -setPoints(int myPoints, int enemyPoints): void -enemyPassed(): void -resetEnemyPassed(): void -passed(): void -resetPassed(): void +log(String text): void -clearLog(): void | |

31. ábra GUI.Table

3.7 Tesztelés terve és eredmények

A tesztelés két részből áll, unit tesztekből és tesztforgatókönyvekből. Ezek Tronfoglalo/test mappán belül található almappákban vannak.

A unit tesztek a játéktábla logikáját, a kártyák adatbázisát és statisztikai adatbázist fedik le. A játéktábla tesztelését a TableTest valósítja meg. Ez a különböző kártyák hatásait, a pontok megfelelő összeadását, kártyák törlését és az AI által használt tryCard függvényt tesztelik, ezzel együtt a cards adatbázis is egy része is tesztelésre kerül.

A TableTest osztályban található tesztek:

- oneCardTest: minden kártyát tesztel, hogy a táblára lerakva jó értéket kapnak-e
- allCardsTest: minden kártyát lerak a táblára és teszteli jó eredményt kap-e
- plagueCardTest: a Járvány kártyát teszteli
- fogCardsTest: a Kød lapot teszteli
- frostCardsTest: a Fagy lapot teszteli
- deleteCardsTest: a lapok törlését teszteli
- tryCardsTest: a tryCard függvény által visszaadott értéket teszteli

A kártyákat tároló adatbázisokat a CardsTest teszteli. Ez teszteli a játékos létrehozását, lapainak lekérését, a játékos alaphelyzetbe állítását, a pakli és további kártyák közötti mozgatást és a kártya adatainak lekérését.

A CardsTest osztályban található tesztek:

- a_playerTest: a játékos létrehozását teszteli
- b_moveTest: minden lapot a pakli helyez, majd vissza a többi laphoz
- c_resetPlayerTest: a játékos alaphelyzetbe kerülését teszteli
- d_getCardTest: három különböző lap adatait kéri le és vizsgálja meg megfelelők-e

A statisztikai adatbázist a StatsTest teszteli. Ez az adatbázis létrehozását, az adatok feltöltését, valamint az AI által használt getChance függvény visszatérési értékét teszteli.

A StatsTest osztályban található tesztek:

- `a_initTest`: az adatbázis létrehozását teszteli
- `b_threeStatTest`: az adatbázist három adattal tölti fel, ebben az esetben 0 esélyt kell visszaadnia
- `c_fourthStatTest`: negyedik adat hozzáadása után már valós esélyt ad vissza, ebben az esetben az átlag pont a maximum és minimum fele
- `d_avgBiggerHalfTest`: az átlag nagyobb, mint a maximum és minimum fele
- `e_avgSmallerHalfTest`: az átlag kisebb, mint a maximum és minimum fele

A tesztforgatókönyvek a játék grafikus felületét, a szerver és kliens működését tesztelik. A tesztesetek lépésekkel állnak, amelyekhez meg van adva a várt eredmények, ez alapján a teszt lépések megfelelhetnek, ha egyeznek az eredménnyel, illetve elbukhatnak, ha a vártnál eltérő az eredmény.

Játék indítása új felhasználóval

| | Művelet | Várt eredmény | Kapott eredmény |
|---|---|---|-----------------|
| 1 | Indítsuk el a játékot | A bejelentkező képernyő megjelenik | Megfelelt |
| 2 | Írunk be egy új nevet | A szöveg megadható | Megfelelt |
| 3 | Nyomjuk meg az OK gombot | Megjelenik a főmenü | Megfelelt |
| 4 | Kattintsunk a Start SinglePlayer gombra | A gomb nem aktív, a rákattintás nem csinál semmit | Megfelelt |
| 5 | Válasszuk az Edit Deck menüt | Megjelenik az Edit Deck menü, 21 lapunk van, a paklink üres | Megfelelt |

| | | | |
|---|---------------------------------------|--|-----------|
| 6 | Rakjunk legalább 20 lapot a paklinkba | A lapok dupla kattintással és az Add gombbal is hozzá adhatóak | Megfelelt |
| 7 | Lépjünk vissza a főmenübe | A főmenü megjelenik, az indító gombok már aktívak | Megfelelt |
| 8 | Lépjünk ki az Exit gombbal | A játék bezáródik | Megfelelt |

Játék indítása meglévő felhasználóval

| | Művelet | Várt eredmény | Kapott eredmény |
|---|--|---|-----------------|
| 1 | Indítsuk el a játékot | A bejelentkező képernyő megjelenik | Megfelelt |
| 2 | Nyomjuk az OK gombra | A főmenü nem jelenik meg, újra megjelenik a bejelentkező képernyő | Megfelelt |
| 3 | Írjuk be az előzőleg használt felhasználó nevet | A főmenü megjelenik | Megfelelt |
| 4 | Kattintsunk a Start SinglePlayer gombra | Megjelenik a térkép | Megfelelt |
| 5 | Lépjünk vissza a jobb alsó sarokban található Back gombbal | Visszakerültünk a főmenübe | Megfelelt |
| 6 | Válasszuk az Edit Deck menüt | Megjelenik a menü, a paklinkban az általunk berakott kártyák vannak | Megfelelt |
| 7 | Lépjünk ki, a jobb felső X gombbal | A játék bezáródik | Megfelelt |

Többjátékos mód tesztelése

| | Művelet | Várt eredmény | Kapott eredmény |
|---|--|---|-----------------|
| 1 | Indítsuk el a játékot, ha még nincs rakunk legalább 20 lapot a paklinkba | A játék elindul, a név megadható, a pakli szerkeszthető | Megfelelt |
| 2 | Válasszuk a Start Multiplayer gombot | Megjelenik a játéktábla | Megfelelt |
| 3 | Az 1. lépés szerint indítsuk el a játékot még egyszer, most más nevet válasszunk | A játék elindul, a név megadható | Megfelelt |
| 4 | Válasszuk a Join Multiplayer gombot | Megjelenik a játéktábla, az egyik játékos kezdhet | Megfelelt |
| 5 | A kezdő játékossal válasszunk egy lapot | A lap lerakható, minden játékosnál megjelenik | Megfelelt |
| 6 | Játsszunk a kör végéig | A kevesebb ponttal rendelkező játékos életet veszt | Megfelelt |
| 7 | Játsszunk amíg az egyik játékos életei elfogynak | A játék véget ér, jó játékos nevét írja ki, a főmenübe kerülünk | Megfelelt |
| 8 | Ismételjük meg a 2,4,5,6,7 lépéseket anélkül, hogy a játékokat bezárjuk | Második játékokra is jól kell működnie a programnak | Megfelelt |

Kapcsolat bontás tesztelése

| | Művelet | Várt eredmény | Kapott eredmény |
|---|---|--|-----------------|
| 1 | Indítsuk el a játékot, ha még nincs rakjunk legalább 20 lapot a paklinkba | A játék elindul, a név megadható, a pakli szerkeszthető | Megfelelt |
| 2 | Válasszuk a Start Multiplayer gombot | Megjelenik a játéktábla | Megfelelt |
| 3 | Az 1. lépés szerint indítsuk el a játékot még egyszer, most más nevet válasszunk | A játék elindul, a név megadható | Megfelelt |
| 4 | Válasszuk a Join Multiplayer gombot | Megjelenik a játéktábla, az egyik játékos kezdhet | Megfelelt |
| 5 | A kezdő játékossal válasszunk egy lapot | A lap lerakható, minden játékosnál megjelenik | Megfelelt |
| 6 | Nyomjuk meg az egyik játékossal az Exit gombot | A játék a főmenübe lép | Megfelelt |
| 7 | A másik játékosnál megjelenik a játék vége ablak, ha nem válasszunk egy lapot vagy passzoljuk | Az ablak megjelenik, ha nem akkor a műveletünk után jelenik csak meg | Megfelelt |

Egyjátékos mód tesztelése

| | Művelet | Várt eredmény | Kapott eredmény |
|---|---|---|-----------------|
| 1 | Indítsuk el a játékot, ha még nincs rakjunk legalább 20 lapot a paklinkba | A játék elindul, a név megadható, a pakli szerkeszthető | Megfelelt |
| 2 | Válasszuk a Start SinglePlayer gombot | Megjelenik a térkép, az első pálya háttere zöld | Megfelelt |

| | | | |
|---|--|---|-----------|
| 3 | Próbáljuk ki a nehézségi szint választó gombokat | Csak egy gomb lehet aktív egyszerre | Megfelelt |
| 4 | Válasszunk egy nehézséget, indítsuk el a játékot | A játék elindítható, a tábla és kártyák megjelennek | Megfelelt |
| 5 | Válasszunk egy lapot, és rakjuk le | A lap lerakható, a táblán megjelenik | Megfelelt |
| 6 | Játsszuk végig a kört | A kör véget ér, ha minden játékos passzolt, a kevesebb ponttal rendelkező játékos elveszít egy életet | Megfelelt |
| 7 | Játszunk a játék végéig | A játék véget ér, a megfelelő embert írja ki nyertesnek | Megfelelt |

Az egyjátékos módban való nyerés és vesztés tesztelése

| | Művelet | Várt eredmény | Kapott eredmény |
|---|--|---|-----------------|
| 1 | Indítsuk el a játékot, ha még nincs rakunk legalább 20 lapot a paklinkba | A játék elindul, a név megadható, a pakli szerkeszthető | Megfelelt |
| 2 | Válasszuk a Start SinglePlayer gombot | Megjelenik a térkép, az első pálya háttere zöld | Megfelelt |
| 3 | Válasszunk egy nehézséget, indítsuk el a játékot | A játék elindítható, a tábla és kártyák megjelennek | Megfelelt |
| 4 | Válasszunk egy lapot, és rakjuk le | A lap lerakható, a táblán megjelenik | Megfelelt |
| 5 | Játsszuk végig a kört | A kör véget ér, ha minden játékos passzolt, a kevesebb ponttal rendelkező játékos elveszít egy életet | Megfelelt |
| 6 | Játszunk a játék végéig, és nyerjük meg (ezt lehet többször kell) | A játék véget ér, a megfelelő embert írja ki nyertesnek, azaz | Megfelelt |

| | | | |
|---|--|---|-----------|
| | próbálni) | a játékest | |
| 7 | Lépjünk be az Edit Deck menübe | Megjelentek az új lapjaink, ezek hozzáadhatók a paklihoz | Megfelelt |
| 8 | Játszunk a játék végéig, és veszítsünk | A játék véget ér, a megfelelő embert írja ki nyertesnek, azaz a gépet | Megfelelt |
| 9 | Lépjünk be az Edit Deck menübe | Ha volt még lapunk a paklin kívül, azok közül vesz el a játék, ha nem volt akkor a paklinkból | Megfelelt |

A mestersége intelligencia tesztelése

| | Művelet | Várt eredmény | Kapott eredmény |
|---|--|---|-----------------|
| 1 | Indítsuk el a játékot, ha még nincs rakunk legalább 20 lapot a paklinkba | A játék elindul, a név megadható, a pakli szerkeszthető | Megfelelt |
| 2 | Válasszuk a Start SinglePlayer gombot | Megjelenik a térkép, az első pálya háttere zöld | Megfelelt |
| 3 | Válasszuk a könnyű nehézséget | A játék elindítható, a tábla és kártyák megjelennek | Megfelelt |
| 4 | Játsszuk végig a játékot | Az ellenfél nem használ különleges lapokat, amíg van más lapja, nem rak szándékosan kicsi lapokat akkor is ha azzal vesztésre áll | Megfelelt |
| 5 | Válasszuk a közepes nehézséget | A játék elindítható, a tábla és kártyák megjelennek | Megfelelt |
| 6 | Játsszuk végig a játékot, legalább 3-szor | Az ellenfél nem használ különleges lapokat, amíg van más lapja, rakhat kis értékű lapokat akkor is ha nem áll vele nyerésre | Megfelelt |

| | | | |
|---|---|--|-----------|
| 5 | Válasszuk a nehéz nehézséget | A játék elindítható, a tábla és kártyák megjelennek | Megfelelt |
| 6 | Játsszuk végig a játékot, legalább 3-szor | Az ellenfél használhat különleges lapokat, rakhat kis értékű lapokat akkor is ha nem áll vele nyerésre | Megfelelt |

4. Felhasznált irodalom

[1] CD Projekt Red: The Witcher 3: Wild Hunt, 2015,

<https://thewitcher.com/en/witcher3>

[2] CD Projekt Red: Gwent: The Witcher Card Game, 2018,

<https://www.playgwent.com/en>

[3] Joshua Bloch: Effective Java, Addison-Wesley, 2001, [346], ISBN:0321356683