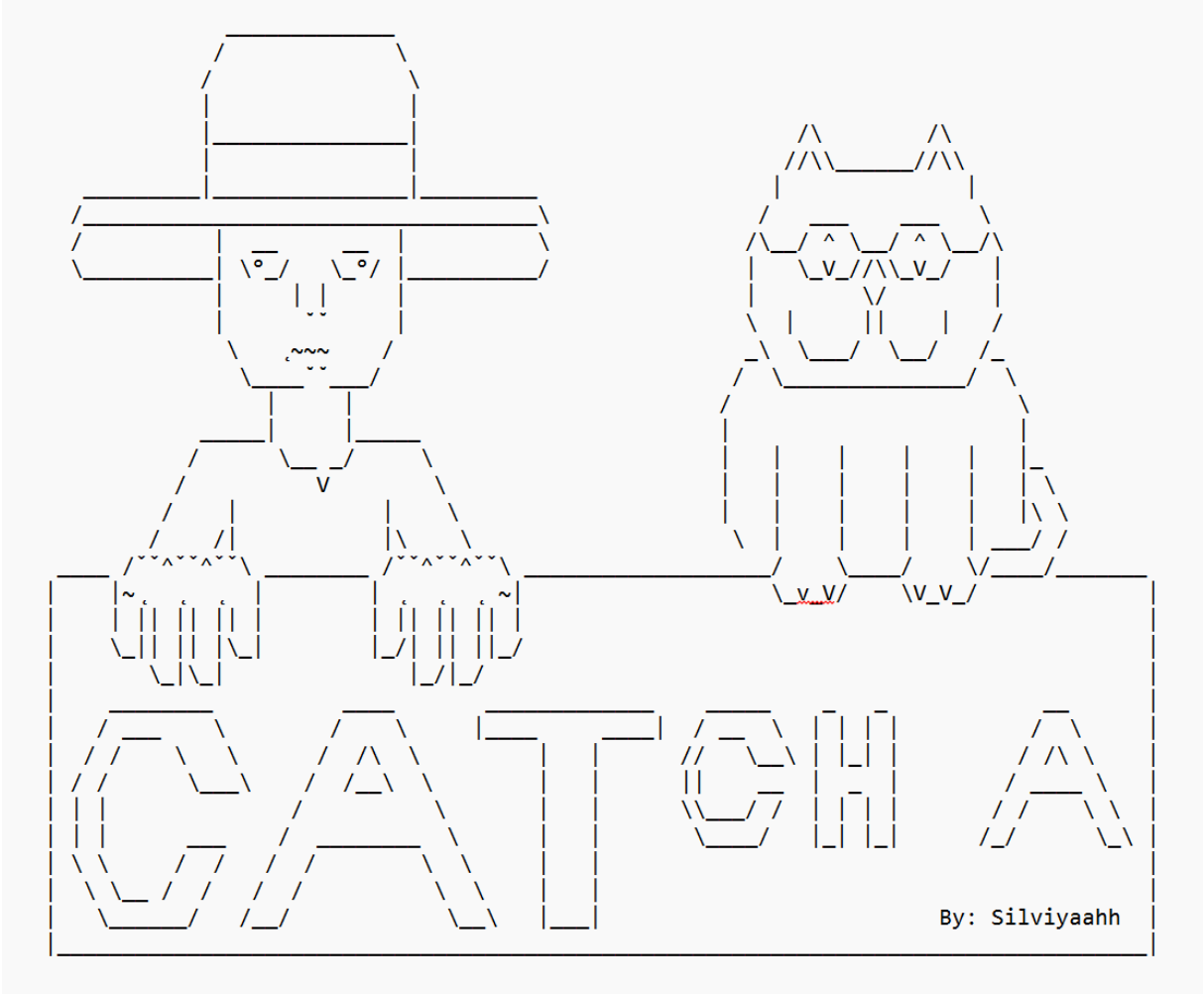
Programozás Alapjai 2

Nagy Házi Feladat

Dokumentáció



Az énáltalam választott házi feladat egy játék elkészítése.

Általános Információk

A játék neve: Catch The Cat (magyarra fordítva: Kerítsd be a Macskát).

A játék típusa: Multiplayer

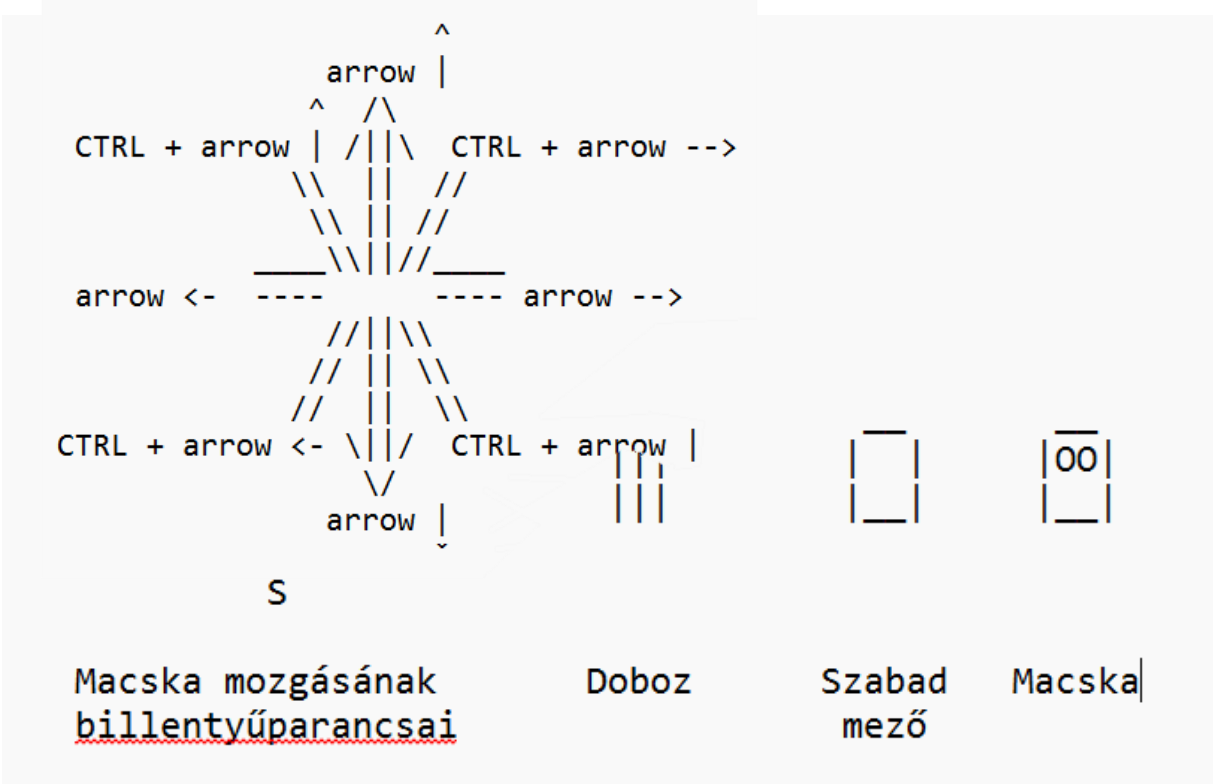
A játék célja: „Gazda” játékos részéről a macska bekerítése; „Macska” játékos részéről a Játéktábláról való kijutás.

A Játék menete: Először beregisztrál/bejelentkezik mindkét játékos, bejelentkezés után kiválasztják, hogy ki szeretne macska lenni és ki szeretne gazda lenni. Ezután létrehozzák a Játéktáblát.

A Játéktábla négyzet alakú mezőkből áll, tetszőleges méretű lehet (ezt a játékos tervezi meg). 3 féle mező létezik: a szabad mezők azok, ahova lehet követ rakni, illetve léphet rájuk a macska. Vannak viszont „kövek” vagy „dobozok”, amelyekre nem léphet a macska. Ezeknek a mezőknek a karaktergrafikus kijelzése a lenti ábrán látható. A játéktábla kezdetben is már tartalmaz néhány ilyen követ tetszőleges helyeken.

Ezután a gazda és a macska felváltva lépnek. A gazda a körében letesz egy követ annak érvényes koordinátáinak megadásával egy szabad mezőre. A következő körben a macska lép. Szabályok: a macska mindig az aktuális pozíciójának megfelelően lép a nyilak vagy a CTRL + nyilak egyikével a lenti ábrán található módon, a tőle körben elhelyezkedő 8 mező egyikére, amennyiben azok szabad mezők, tehát nincs rajta doboz. A Macska nem tud átugorni egy vagy több mezőt, sem dobozt, és nem tud olyan mezőre lépni, ahol már doboz van.

A játéknak akkor van vége, ha a macska nem tud semerre mozogni (a körülötte lévő mind a 8 mezőn doboz van), vagy amennyiben a macskának sikerült elérnie a pálya bármely szélét. Az első esetben a gazda számára nyereséggel, a macska számára veszteséggel zárul a játék, míg ellenkező esetben értelemszerűen fordítva.



A játék további funkciói: A befejezett játékok adatait egy txt fájlba menti a program: a játékosok kivoltát, a játék végállapotát, valamint a tábla méretét. Ezeket felhasználók szerint lehet szűrni: egy játékos bejelentkezés után láthatja, hogy mely játékokat játszotta és kikkel.

Mindemellett értelemszerűen a felhasználók becenév-jelszó párosát is menti egy külön fájlba e-mail címükkel együtt, valamint, hogy hány játékot nyert az adott felhasználó összesen és abból mennyit nyert meg. Ugyanitt lehetőség van a profilod módosítására: felhasználónév vagy jelszó megváltoztatására, valamint elfelejtett jelszó javítására e-mailes azonosítással.

Újabb funkció lesz a játék értékelése: szöveges véleményt írhat a játékos, pontozhatja 1-től 5 csillagig a játékot, valamint lehetőség van mások kommentjeinek, értékeléseinek megtekintésére is. A pontozásokból elért átlag értékelést (az összes értékelések számának függvényében) megjelenítjük a kezdőképernyőben.

A játék teljes egészében karaktergrafikus megjelenítéssel fog működni, valamint enterek használata nélkül, tisztán gombokkal (nyilakkal, tabulátorral, Escape karakterrel, Delete karakterrel, vagy ezek Ctrl gombbal történő egyesítésével).

Felhasználói Interakciók

A játék angol nyelven működik, azonban a funkciókat magyar fordítással fogom részletezni. A játék kezdőlapja az a kép, amelyet a dokumentáció elejére is beszúrtam: erre bármely gombot megnyomva egy, a fel-le nyilakra váltakozó menüt látunk magunk előtt, melynek elemei:

- Új játék
- Mentett játék betöltése
- Mentett végeredmény kirajzolása
- Vélemények
- Értékelj minket
- Profilmódosítás
- LASAGNAAAA

A „LASAGNAAAA” menüpont egy másfél-kétperces rövid, ám vicces animációt vetít le, mely végén megtudhatjuk, hogy készül az igazi olasz Lasagna. Ugancsak bármely gombot megnyomva visszatérünk a főmenühöz.

A „Profilmódosítás” menüpontot választva be kell jelentkezni azzal a profillal, amit módosítani szeretnél, majd a megjelenő menüből a „becenév módosítása” vagy a „jelszómódosítás” menüpontot választva a kívánt funkció értelemszerűen teljesül, miután a felhasználó e-mail-ben azonosította magát.

Az „Értékelj minket” menüpontot választva a megfelelő funkcióhoz jutunk: be kell jelentkezni, be kell írni egy 1-től 5-ig terjedő egész számot, hogy hány csillagra értékeled a játékot, majd opcionálisan írhatsz hozzá szöveges megjegyzést is. Ezután visszatérünk a főmenübe. Fontos azonban, hogy csak akkor írhat értékelést a felhasználó, hogyha már legalább 1 játékot elkezdett.

A „Vélemények” menüpont egészen egyszerűen egyesével visszaírja a txt fájlba mentett értékelés adatokat, melyek között a felhasználó a jobbra-balra nyilakkal mozoghat. Az értékeléssel ellentétben itt nem kell bejelentkezni.

A „Mentett játék végeredmény kirajzolása” menüpont alatt ismét be kell jelentkezni, sikeres bejelentkezés után pedig a mentett játéktáblák között lehet lépkedni ugyancsak a jobbra-balra nyilakkal. Az adott felhasználó által elmentett játék kimenetelek mindegyike megjelenik, függetlenül a játékos aktuális játékban való szerepétől és partnerétől. Innen és az egyel korábbi funkcióból az Escape gomb megnyomásával lehet visszalépni a főmenübe.

„Mentett játék beolvasása” esetén mind a két játékosnak be kell jelentkeznie szerepek szerint rendben, majd a mentett játékok közül szintén ugyanúgy lehet váltogatni, betöltés pedig az Enter megnyomására történik. Innen tovább folytatódik a játék annak rendje módja szerint.

„Új játék” esetén is két játékosnak kell bejelentkeznie szerephelyesen (először macska utána gazda), azonban itt már regisztrálhat is új felhasználó.

Regisztráció esetén először választani kell egy egyedi felhasználónevet. Ezután meg kell adnod az e-mail címedet. A program küld egy e-mailt a megadott e-mail fiókra egy generált kóddal, amit helyesen visszaírván hitelesítetted az e-mail címedet (a továbbiakban: e-mailes azonosítás). Ezután ki kell találni és meg kell erősíteni a jelszót.

Bejelentkezés során pedig bekéri az e-mail címet és a jelszót, helytelen pár esetén pedig választhatsz, hogy jelszót módosítasz, vagy újra megpróbálsz bejelentkezni.

Ha mindkét játékos bejelentkezett a játék sztoriját tovább tekerve létre kell hozni a pálya alakját. Ehhez meg kell adni a pálya kívánt maximális szélességét és magasságát.

A játék további menete az *Általános Információk* bekezdés *Játék menete* szakasza alatt olvasható részletesen. Mindaddig, amíg cica esetén a CTRL + Delete betűkombinációval, macska esetén az „F11” koordinátával ki nem lépünk (ilyenkor megkérdezi, akarjuk-e elmenteni a játékot) ezen az alapelven folyik tovább. Játék vége esetén bármely gomb megnyomására visszatérünk a főmenübe.

Hibakezelés

Felhasználó általi bemeneti hibából a felhasználó gyakorlatilag semmit nem érzékel: a helytelenül beírt adatot a program kitörli, és ugyanoda kell beírni újból egészen addig, amíg helyes adatot ír be a felhasználó

Adattárolás

Az adatok tárolását 5 db, egymással összefüggő osztály és dinamikus tömbök, valamint még 3, az előzőktől független, azonban egymástól függő segéd osztály segítségével fogom tárolni.

User osztály

Minden felhasználóról tároljuk a becenevét (*nickname*), jelszavát (*password*), *e-mail cím*ét, valamint, hogy hány játékot nyert (*games*). Konstruktorát a globális register() valamint login() függvények hívják meg, melyek létrehozzák a játékos objektumát. Profilmódosításhoz használunk továbbá setter fügvényeket is.

Önmagában egy ilyen objektum elég ahhoz, hogy azzal a játékos írni tudjon értékelést, módosítsa a profilját, vagy a saját kész játékainak végső állapotát lekérje, azonban, mivel ez egy multiplayer játék, ahol a két játékos teljesen máshogy viselkedik, így ennyi önmagában nem elég ahhoz, hogy a játékot lebonyolítsuk. Erre szolgál a **Cat** és a **Man** osztály, melyek miatt létre kellett hozni a jelen tárgyalt osztályba egy virtuális lep() tagfüggvényt, melynek jelentését az előbbi két másik osztálynál tárgyalom bővebben.

Man osztály

A gazda osztálya megörökli a **User** osztályt, így tehát használhatja annak minden publikus tagfüggvényét. Más egyéb privát adattagot nem kap, azonban itt definiálom a lép() függvényt, mely a játék során lebonyolítja a gazda lépéseit, azaz bekér egy érvényes koordinátát a játéktábláról (pl.: A3), majd, ha az a mező üres, a **Mező** osztályban részletezett módon követ tesz le a mezőre, amennyiben pedig helytelen vagy nem szabad mező koordinátája érkezik be újra kéri egészen addig, ameddig helyes koordinátát nem kap.

Cat osztály

A cica ugyanúgy megörökli a **User** osztályt, hiszen az ő esetében is egy játékosról beszélünk, azonban mozgása és feladata jelentősen eltér a **Gazda** feladatától, ebből kifolyólag kapott új osztályt. Az öröklött adatokon kívül kap egy **Mező** típusú privát adattagot is, mely arra a mezőre mutat a játéktáblán, amelyen a macska éppen aktuálisan áll. Ennélfogva tehát a részletezett osztályunk tartalmazza a **Mező** osztályt, hiszen, ha a mező megszűnik, a macska is megszűnik.

Lépésének algoritmusa pedig a következő: először is ellenőrzi, hogy éppen a szélén áll-e a táblának, illetve ezután a nyilak vagy Ctrl + nyilak gombok valamelyikét várja a felhasználótól az *Általános Információk* bekezdés *Játék menete* szakasza alatti ábrán jelzett

jelentéstartalommal. Az így meghatározni kívánt mezőt ellenőrzi, hogy szabad mező-e, amennyiben igen, úgy az aktuális helyzetét megváltoztatja a kívánt mezőre. Amennyiben nem, vagy más hibába ütközünk, újra kéri a felhasználótól egészen addig, amíg helyesen tud mozogni. A helyzetváltásban nagy segítséget jelentenek a **Mező** osztály további adattagjai. Azt, hogy van e lehetséges lépése a macskának a játék függvény ellenőrzi

Utolsó tagfüggvénye a drawline1() és drawline2(). Ezek rajzoló függvények, abban segítenek, hogy az *Általános Információk* bekezdésben látható karaktergrafikát ki tudja nekünk írni.

Mező osztály

Minden bizonnyal a projekt egyik legkomplexebb osztálya. Egyrészt van egy koordinátáit tároló, **Pont** típusú adattagja, tartalmazza tehát a **Pont** osztályt (ha a Pont megszűnik, a mező is). Tárol továbbá egy egyedi beszédes nevet a mezőnek a további könyebb azonosítás érdekében, illetve logikai típusú adattagként tárolja, hogy van-e rajta kő, szélén van-e, létezik-e a mező (pálya része-e), szabad-e rálépni. Ez utolsót azért kell külön emelni a kőtől, mert bár soha nem szabad, amikor kő van rajta, de nem mindig van rajta kő amikor szabad (a macskára nem szabad rátenni követ). Konstruktorai között említésre méltó a paraméter nélküli konstruktora. Ez ugyan létrehoz egy logikailag helyesen létező mező, azonban a -1:-1 koordinátával, valamint „INVALID” ID-vel rendelkezik. Tagfüggvényei közé tartozik a kő lerakása (megfelelő adattag beállítása) és a macskához hasonló kirajzol1() kirajzol2() függvényei, valamint a szabad és a széle adattaghoz létezik egy setter függvény. Két mező összehasonlítása érdekében tartalmaz egy == operátort.

Kiolvasando osztály

A játékban kétféle különböző dolgot is fájlból kell kiolvasni úgy, hogy a jobbra-balra nyilakkal kelljen az elemei közt váltogatni: a táblát és a kommenteket. Ezeket azonban teljesen különböző módon lehet kirajzolni, mindegyiknek más része van. Ehhez segítségként hoztam létre ezt az osztályt. Az említett függvény egy Heterogén kollekciót kap meg, melynek az elemiet olvassa ki. A kollekció közös osztálya a mostani. Egyetlen adattagja nem csinál semmi hasznosat, csak azért van ott, hogy legyen adattagja, ezenkívül virtuális függvénye a kirajzol, melyet a leszármazott osztályokban definiálok.

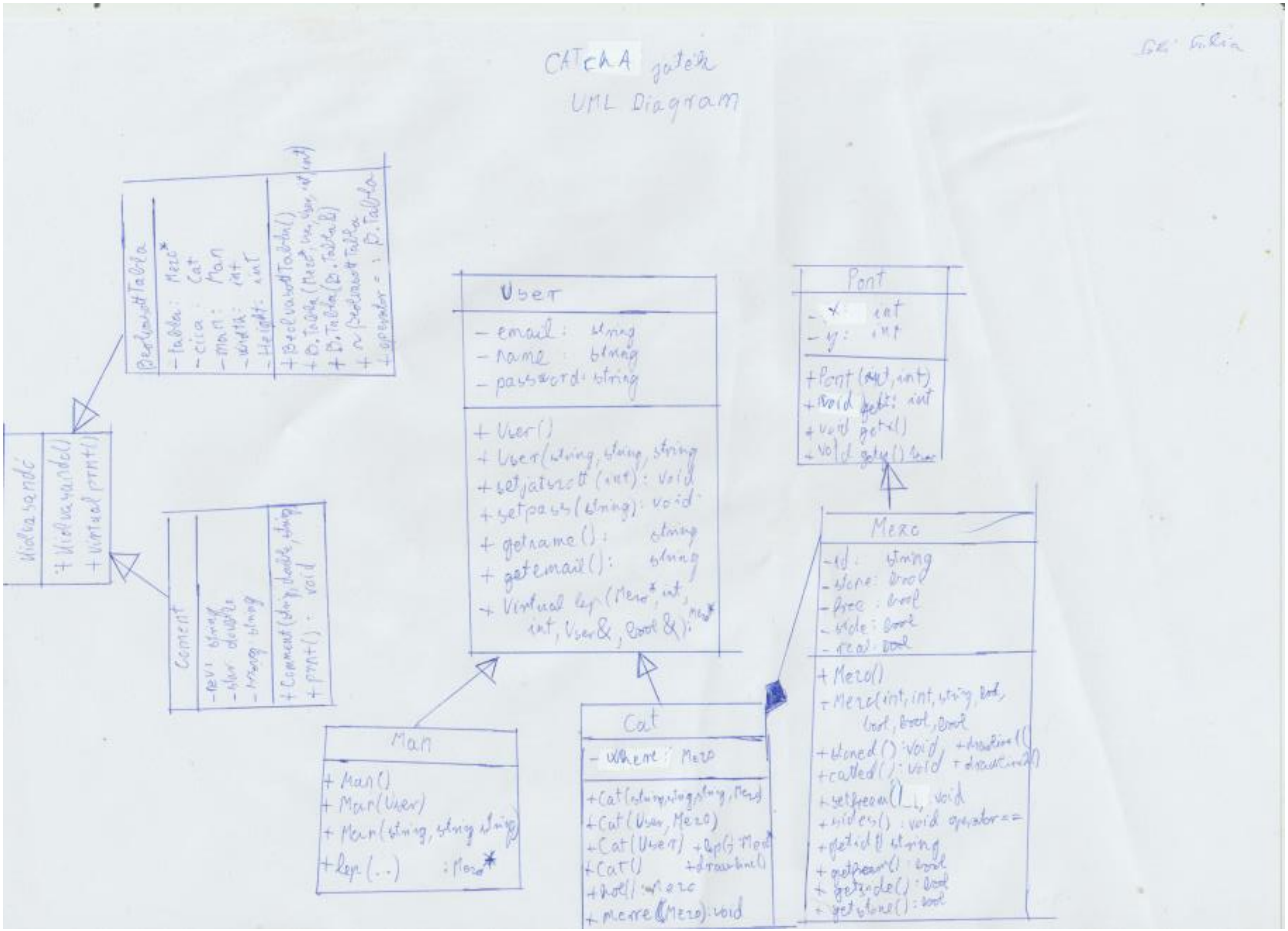
Comment osztály

A kommenteket tárolja, a Kiolvasando osztály leszármazottja. Adattagjai az adott komment értelmi szerzőjének felhasználónevét, szöveges és numerikus értékelését tartalmazzák. Kiíró prnt() függvényét a függvények legelején részletezem.

BeolvasottTabla osztály

A korábban elmentett játékok objektuma. Benne van minden, amit elmentettünk a játékban: a macska, a gazda, a tábla dinamikusan foglalt Mezok tömbjével. prnt() tagfüggvényét úgyszintén a függvényeknél részletezem.

Az UML diagram a következő oldalon található


UML Diagram
User
- email: string
- name: string
- password: string
+ User()
+ User(string, string, string)
+ setpass(string): void
+ getname(): string
+ getemail(): string
Pont
- x: int
- y: int
+ Pont(int, int)
Mezo
+ Mezo()
Cat
+ Cat(User, Mezo)
+ Cat(User)
+ Cat()
Man
+ Man()
+ Man(User)
Coment
+ print(): void

Függvények

Comment::prnt()

Kiírja az adott kommentet. A kommentnek 3 része van: a felhasználóneve, az hogy hány csillagot adott rá és a szöveges értékelés. A nevet aláhúzza, a csillagokat kirajzolja az alább részletezett csillag() függvénnyel, majd aláírja a szöveges értékelést.

BeolvasottTabla::prnt()

Kiírja a két játékos felhasználónevét alatta pedig kirajzolja a táblát.

Man::lep(Mezo* tabla, int width, int length, User& masik, bool& gameover)

Az emberke lépése abból áll, hogy kiválaszt egy tetszőleges szabad cellát, és belerak egy követ, ezzel elérvén, hogy a macska ne tudjon arra a mezőre lépni már innentől kezdve. Ezt pedig teszi úgy, hogy beírja a kiválasztott mező koordinátáit. Vigyázat: a megszokott koordináta-átadástól eltérőn ebben az esetben nem az oszlop, hanem a sor koordintája kell először. Ezután ellenőrzi a függvény, hogy létezik-e egyáltalán olyan koordinátjú mező, vagy ha létezik, akkor van-e már rajta eredetileg is kő vagy macska. Egészen addig írhatja be a játékos a kódot, míg nem rakott le legalább egyszer helyesen követ a körben.

Cat::lep(Mezo* tabla, int width, int length, User& masik, bool& gameover)

A macska a fenti játékleírás szerint lép a nyilak és a Ctrl + nyilak segítségével, valamint kezeli azt, hogy ha a macska ki akarna lépni a játékból. Elsőzöris meghatározza, hogy merre akar mozogni a játékos. Meghatározza, hogy helyes mezőre akar lépni, amennyiben nem, addig próbál újra lépni a macska, amíg nem tud lépni. Továbbá kezeli azt is, hogy ha nyilak/Ctrl + nyilak helyett Ctrl + Delete-t nyom, akkor kilép. Helyes kilépés esetén a játékból is kilépnek a játékosok, ezért kapják meg a gameover változónak a referenciáját. Ez a változó irányítja ugyanis a játék magjának a ciklusát, hogy meddig lépjenek egymás után az egyes játékosok. Az ezeket a funkciókat megvalósító függvények leírása a továbbiakban olvasható.

Fajlkiolvas()

Ez egy egyszerű általános célú függvény, egy txt fájl tartalmát olvassa ki betűről betűre. Nem tárolja a tartalmát, csak egy az egyben kiírja a képernyőre.

Sajatatoi(char ch)

A mezők indexeléséhez van köze. A Mezők koordinátáját találja ki a beírt ID-ból: az egyes mezők ID-je úgy áll össze, hogy a Mező x és y kooridinátájához rendel egy betűt és összevegyíti. Ez a függvény ennek a fordítottját csinálja: kitalálja a mező ID-jének egy betűje alapján, hogy az milyen koordinátához tartozott.

Osszesito()

Egy osszesito.txt fájlba minden komment megírása után frissül, hogy hány kommentet írtak összesen, és ebből átlag hány csillagra értékelték a játékot. A függvény ebből a fájlból olvassa ki ezeket az adatokat és el is tárolja.

Valaszto(int elsovalasz, int maxvalasz, int jelenlegi, const char* fajlnev, int ertekeles) és arrowfind(int elsovalasz, int maxvalasz, const char* fajlnev, int ertekeles)

Ha több lehetőség közül lehet választani a fel-le nyilak segítségével, annak a logikáját tartalmazza. Az arrowfind a mozgási logika, amely leírja, hogy melyik gomb megnyomására hogy változzon az index hogy a lehetőségek közül melyiket választja a játékos, míg a valaszto pontosíja az indexet, kirajzolja a választási lehetőségeke tartalmazó fájlt, majd egy nyíllal jelöli, hogy a játékos éppen melyik lehetőségen áll.

csillag(double star, double needquestion)

A csillag.txt fájl egy csillagot tartalmaz. A függvény megkap egy double értéket, hogy hány csillagot kell kirajzolnia, majd tizedpontossággal kirajzol ugyanennyi csillagot. Mivel pontosan 10 karakterből áll a csillag, így minden tized értékhez tartozik 1 karakternyi csillag. A needquestion azt jelöli, hogy szükséges-e megkérdezni a játékost hogy hány csillagot akar (részletek az ertekel() és a walkcomments() alatt).

animate()

A rövid animációt megvalósító program: 2 másodpercenként automatikusan kirajzolja az animáció következő jelentét, melyek hollétét az animation.txt fájlból olvassa ki.

bipolar(std::string paratlan, std::string paros)

Ha 2 válaszlehetőség között kell választani, azt a tabulátorral lehet megtenni. Ez annak a függvénye: megkapja a két válaszlehetőséget tartalmazó txt fájlt, majd tabulátor egnyomásának hatására vált a kettő között, enter megnyomására visszaadja, hogy melyiket választotta a játékos.

modositFelhasznalo(const std::string& fajlNev, const std::string& becenev, int funkcio, const std::string& ujErtek) és usermodosit() és jelszomodositas(User user)

A felhasználók módosítását megvalósító függvény. Az arrowfind (és így a valaszto) függvnyeket meghívva kiválasztatja a játékossal, hogy mit akar módosítani, előkeresi felhasználónév alapján az e-mail címet, küld rá egy véletlenszerű kódot, majd hogyha a felhasználó sikeresen azonosította magát a játékosok adatait tartalmazó játékos.txt fájlban megváltoztatja a megfelelő értéket az újonnan bekértre. A fájlkezelést belőle a modositFelhasznalo végzi el, mely funkcio paraméterében megkapja, hogy melyik értéket kell módosítani (1-2-3 – jelszó-email-megnyert játékok száma), míg a logikai megvalósításért a usermodosit() felel. A jatekos.txt fájl tartalmát a regisztrációs függvény leírásánál részletezem. A jelszó módosítása azért van külön függvénybe szervezve, mert azt nem csak a felhasználó módosítása menüpontból érhetjük el, hanem bejelentkezéskor hibás jelszó beírása esetén

keresEmailCim(const std::string& fajlNev, const std::string& becenev)

A felhasználókat tartalmazó fájlban megkeresi az adott névhez tartozó email címet. A fájl kinézetét a regisztrációs függvénynél részletezem.

mail(std::string cim, std::string body)

egy megadott e-mail címre e-mailt küld a saját gmail fiókomról body szöveggel és „CATchA email verification) tárggyal és egy véletlenszerűen generált kóddal egy c++ kódból meghívott powerShell paranccsal. Visszatér a véletlenszerűen generált kóddal.

vanuser(const std::string keresett)

ellenőrzi, hogy a megadott felhasználónév szerepel-e már a felhasználókat tároló fájlban, majd visszatér ennek a logikai értékével. Mivel a felhasználókat a felhasználónevük alapján azonosítjuk, fontos, hogy mindenképpen egyediek legyenek.

ervenyes_felhasznalo(const std::string& fajlnev, const std::string& becenev, const std::string& jelszo)

Végigmegy a már felregisztrált játékosok listáján, és ellenőrzi, hogy a megadott felhasználónévhez érvényes jelszót adtunk-e meg. A feltételes ágnak köszönhetően akkor is hamis logikai értéket ad vissza, ha nincs megadott nevű játékos elmentve a fájlban.

szovegbekeres(std::string fajlnev)

A paraméterként megkapott fájlban lévő kérdést teszi fel a játékosnak, amennyiben arra szöveges váaszt várunk, és visszatér a válaszunkkal.

verifymail(std::string email, int counter, std::string body)

a paraméterkét megkapott emailt ellenőrzi: megpróbál emailt küldeni rá, siker esetén bekéri a kódot és ellenőrzi, hogy helyesen írta-e be a játékos. Amennyiben nem sikerült eltalálni, 3 próbálkozás után újra kell írni az e-mail címet, a counter azt számolja, hogy hány lehetőség van még addig, amíg újra nem kell írni. Amennyiben nem sikerült e-mailt küldni, úgy automatikusan újra kell írni az e-mail címet.

Regisztracio()

```
e-mail cím: soki.szilvia03@gmail.com
Becenév: WonderGurl
Jelszo: StrongPassword
Jatszott: 0

e-mail cím: soki.szilvia03@gmail.com
Becenév: PearL
Jelszo: NotTooStrong
Jatszott: 0

e-mail cím: szilvi.egyetemi@gmail.com
Becenév: pearfusion
Jelszo: NaturAqua
```

A felhasználótól bekért adaokat és azok sorrendjét a játék mködésének leírása során már részleteztem. Az alapján kéri be az adatokat, szükség esetén a megfelelő funkció működéséért felelős függvény meghívása segítségével, majd a kapott adatokat a jatekos.txt fájlba menti el a következő formátumban: (balra egy részlet a fájlból):

Bejelentkezes()

Menetét a játék leírásánál részleteztem. A megfelelő ellenőrző függvényeket meghívva ellenőrzi a kapott adatok helyességét, majd sikertelenség esetén választás elé állítja a játékost: újra megpróbálja vagy jelszót módosít, majd annak értelme szerint halad tovább.

leftright(double start, double diff, std::function<void(double, bool)> muvelet, double min, double max, bool funchoz)

Egy min és egy max érték között léptet diff értékkel feljebb vagy leljebb egy indexet attól függően, hogy a játékos a jobb vagy a bal nyilat nyomta meg. Azt, hogy mit csináljon ezen kívül a gombok megnyomására az aktuális értékkel a program, egy funktor adja meg. Visszaadja azt az indexértéket, ahol a játékos megállt, és entert nyomott.

ertekel()

```
LunaBy
4.5 csillag
no

G3rgo5463
5 csillag
Pr□tty vagy mi :'D

G3rgo5463
5.1 csillag
I LOVEDDD it

DinDinDinDun
3 csillag
Im an Orange. Im an Italian b
are watching the comments on
Tung Tung Tung Tung Tung Tung
```

Bejelentkezés után a játékosnak meg kell adnia, hogy hány csillagra értékeli a játékot az 5-ből. Dinamikusan változtatható a csillagok értéke a nyilakkal, 0.1-től 5.1-ig. Minden alkalom után, amint módosul az index, és dinamikusan kirajzolja a csillagokat, miközben felette végig megmarad a kérdés. Ehhez használom a csillag függvény needquestion bemenetét, lesznek ugyanis olyan esetek, amikor nem kell kirajzolni ezt a kérdést a csillaggal együtt, csak a csillagot. A szöveges válasszal együtt pedig elmenti a kommenteket egy velemenyek.txt fájlba a következő formátumban: (egy részlet a lap bal oldalán található a fájl tartalmából)

read_comments(size_t& count, const std::string& filename = "velemenyek.txt")

Kommenteket olvas ki az előbb részletezett velemenyek.txt fájlból, mindegyiket egy új objektumba menti, majd az egészet egy pointereket tartalmazó dinamikus tömbben visszaadja az összeset. A tömböt aszerint allokálom, hogy hány kommentet tudott kiolvasni a fájlból a program, és ezt a count referencia értéknek köszönhetően ezt is vissza tudom adni a kommentekkel együtt

walkcomments(Kiolvasando allcomments, size_t count)**

Megkap egy heterogén kollekciót, melynek az alapja az osztályok között részletezett Kiolvasando osztály pointereiből álló dinamikus tömb, továbbá ennek a mérete. Végigmegy a tömb elemein, és meghívja a prnt() függvényeit amivel kirajzolja a kívánt dolgot, majd, ha a játékos befejezte a nézelődést és az Escape karakterrel jelezte hogy vissza akar lépni a főmenübe, törli a táblát.

sizew() és sizel()

Segédfüggvények, az ujjatek() fogja őket használni. a korábban részletezett leftright() függvényhez tartozó funktor. A tábla méretének bekéréséhez kell, ami ebből sejthetően a nyilakkal változtatható nagyságú, 8-tól 43-ig, illetve 8-tól 18-ig (szélesség, magasság) terjedően.

szemelyek(std::string paratlan, std::string paros, std::string kit, std::string info)

Segédfüggvény. Új játék létrehozásánál ki kell választanunk, hogy ki melyik játékos lesz. Mindkét játékosnak regisztrálnia kell vagy be kell jelentkeznie amennyiben ezt már megtette korábban. Ezt hivatott teljesíteni a függvény, s miután megtette, a játék szereplői bemutatkoznak.

tablarajzol(Mezo* tabla, int width, int length, Cat cica)

Egy játéktáblát rajzol ki. A tábla mezőit dinamikusan foglalt tömbben kapja meg, végigmegy az egész tömbön és az összes elemére meghívja a drawline1() és drawline2() funkciókat, a sorok végén új sort kezdve. Sor végét az indexekről ismer fel: ha az x koordináta eléri a tábla hosszát, akkor sor végén vagyunk és új sort kell kezdeni. A táblán kívül egy rácshálózat is kirajzolásra kerül, amelyről le tudja olvasni a játékos a mezők koordinátáit.

keresszomszed(Mezo* tabla, Mezo mezo, int xdiff, int ydiff, int length, int width) és catstepcorrect(Mezo next)

Egy paramétereként megkapott mezőnek egy konkrét szomszédját keresi meg, figyelembe véve, hogy a sarokköveknek kevesebb szomszédjuk van mint az oldalköveknek vagy a középső köveknek. Meg kell adni továbbá paraméterként, hogy x és y irányba hány mezőnyi elmozdulással keressük a szomszédot, így nem csak közvetlen szomszédok, hanem általánosságban bármilyen, táblában adott ponttol adott elmozdulásra lévő mezők megtalálására is működne a függvény. Amennyiben a megfelelő mező nem létezik, vagy éppen nem szabad

mező, úgy az előre definiált invalid mezőt adja vissza a függvény. Ezzel dolgozik a catstepcorrect függvény, mely a visszaadott mezőt összehasonlítja az előre definiált invalid mezővel, és amennyiben egyezik, sikertelenre állítja a macska lépését.

tablament(std::string fajlnev, Mezo* tabla, int width, int length, Cat cat, Man man) és tablakiolvas(std::string fajlnev)

```
Cat was o
Man was o
Width of table: 5
Height of table: 5
Fields: (id | stone | side | free)
00111
01010
02010
03010
04010
10010
11000
12101
13000
14010
20111
21000
22101
23101
24010
30111
31000
32101
33000
34010
40111
41111
42111
43010
44010

Cat was kiscica
Man was o
```

Elmenti a tábla adatait a paramétereként kapott fájlba illetve olvassa ki onnan a következő formátumban: (balra a fájlból található egy részlet). Utóbbi függvény egy std::vectorba gyűjti össze a kiolvasott adatokat, mely vektor így tehát BeolvasottTabla példányokat tartalmaz a fájlban szereplő összes elmentett táblával.

find_tables_by_username(const std::string& username, size_t& found_count, const std::string& filename = "finished.txt") és tablakinez()

A tablakiolvas függvény által visszaadott BeolvasottTabla példányokat tartalmazó vektort leszűkítve visszaad belőle egy olyan BeolvasottTabla pointereket tartalmazó dinamikusan allokált tömböt, melyben csak azok a játékok vannak benne, melyben a megadott felhasználónévvel rendelkező játékos játszott. Mindezt a tablakinez függvény kontrollálja, kéri be a keresett felhasználónevet és hívja meg a függvényt a megfelelő paraméterekkel.

savechoice(User embi1, User embi2)

Amikor valamelyik játékos ki akar lépni a játékból, akkor azt mindkét játékosnak meg kell erősítenie ezt a döntést, hogy megbizonyosodjunk róla, hogy közös döntés volt. Éppen ezért e-mailes azonosítást kér ilyenkor mindkét játékostól a fent részletezett módon. Amennyiben nem jól írják be a kódot, visszatérnek a játékba, és nem lépnek ki belőle. A függvény tehát egy logikai

értéket ad vissza, és ettől fog függeni, hogy az aktuális tábla elmentődik-e vagy nem (lásd bővebben: Macska::lep.

tableuncat(Mezo* tabla, int width, int length, Mezo now, Mezo next)

A macska mezőjének free flagje hamis, hiszen értelemszerűen a macskára nem rakhatjuk rá a követ. Azonban ha a macska elmegy egy mezővel arrébb, a régi helyére már rakhatunk követ, így lépésnél az eredeti helynek a free flagjét át kell állítani. Pontosan ezt csinálja a függvény. Visszaadja a módosított játéktáblát.

cicanemlephet(Cat cat, Mezo* tabla, int length, int width)

A játék magjául szolgáló függvény hívja meg. Lényege, hogy a macska aktuális pozíciójának mind a 8 lehetséges szomszédján végigmegy, és ha van olyan mező, ahova léphet, akkor igaz logikai értéket ad vissza, ellenkező esetben hamisat. Ez egy végtelen ciklust előz meg, hiszen ha bekerítették a macskát végtelenségig próbálkozhat lépni, nem fog neki sikerülni.

jatsz(Mezo* tabla, Cat cat, Man man, int length, int width) {

Az egész játék magja. Ez irányítja a macskát és az embert, hogy mikor lépjenek stb. Magja egy ciklus, mely addig fut, míg nincs vége a játéknak (gameover változó). Ebben először a macskát léptetik, utána az embert. minden lépéshez tartozik egy magyarázat, hogy ki lép és hogyan, txt fájlokban. Ezután, ha mindketten léptek, a játék ellenőrzi, hogy van e nyerés, vagyis azt, hogy a macska valamelyik szélső mezőn áll-e, avagy van e helye bármerre mozogni. A játék felépítéséből adódóan tehát ha a cica kilépett a tábla egy szélére, akkor még az emberkének le kell raknia még egy követ, és csak ekkor mászik ki Oswald-macska a kerítésen. Amennyiben pedig a játék közben a játékosok úgy döntenek, hogy nem akarnak tovább játszani, ki akarnak lépni és sikerül is, akkor pedig megáll az egész játék, akárki is lépett, és visszatérünk a főmenübe. Nyerés esetén elmentődik a játék jelenlegi állapota és Oswald-macska és Spencer-bácsi is üzen valamit a játékosának. A végén kitörlöm a táblát, amit dinamikusan foglaltam a játéktérnek.

ujjatek()

Amikor egy játékos új játékot kezd, ez a függvény indul el. Bekéri a játékosok kilétét és a tábla méretét, majd elindítja a játékot a kívánt paraméterekkel. Az ezeket megvalósító függvényeket