GRAFIKUS FELÜLET SPECIFIKÁCIÓJA

40 – ZETA

Konzulens:
KOVÁCS BOLDIZSÁR

Csapattagok

Alpek Dávid Zsolt	C31X0F	alpek.david.zsolt@gmail.com
Csia Klaudia Kitti	HA5YCV	kitkat@sch.bme.hu
Litavecz Marcell	IPHJNB	marcell.litavecz@gmail.com
Marton Judit	M0MYIM	judit.marton@edu.bme.hu
Ruskó Eszter	H8IBRC	eszter@rusko.hu

2022. 05. 04.

11. Grafikus felület specifikációja

11.1 A grafikus interfész

11.1.1 Karakter képek

Itt azok a képek találhatóak meg, amelyek több játékos esetén rajzolja ki a különböző színű karaktereket, illetve viselt eszközöknél, hogy fog kinézni a *CharacterView*-ban a karakter, és bónuszként, hogy mivé fog alakulni egy medvetánccal fertőzött karakter. A zsák nem fog látszódni egy ábrán se, ugyanis az nálunk úgy van lekezelve, hogy csak plusz számláló bővítőként van hozzáadva a karakter *Inventory*-jához, így ahhoz nem készült rajz.

Az alap karakterünk, ez lesz a kiinduló verzió.

Ha több játékos is belekerül, akkor ilyen színnel fognak megjelenni a pályán.

Alap karakterünk [balról jobbra] kesztyűvel, fejszével, majd kesztyűvel és fejszével. Ezek a verziók + a köpenyes verziók nem fognak látszódni az alap játékban, hanem kizárólag csak a *CharacterView*-ban, ugyanis alapból azt feltételezzük, hogy a karakterek vakok, így mindig csak az alap ruhát látják egymáson.

A karakterünk [balról jobbra] köpennyel, köpennyel és fejszével, köpennyel és kesztyűvel majd a legvégén mind a három eszközzel együtt.

Így néz ki egy medvetánccal fertőzött játékos, természetesen ez csak mások szemében néz ki így, ugyanis, akit megfertőztek az már csak a Game Over feliratot fogja látni. Másik játékosok szemében azonban egy folyamatosan mozgó, raktárt fosztó, véres-szemű, fertőző medvét fog látni, aki, ha megjelenik egy pályán számolhatunk a problémákkal.

11.1.2 Objektum képek

Itt olyan képek találhatóak meg, melyeket a játékos maga tud eszközölni, például az *Inventory* és *CharacterView*, majd a *SkillBar*.

A játékos *Inventory*-ja a játék kezdetében. Felső részben látható a karakter, ahogy éppen kinéz, és milyen eszközök találhatóak rajta. [Fentről lefele] látható a fejsze, köpeny, kesztyű végül a zsák helye. Az alsó részen meg látható az *Inventory*, amelynek ez a kezdeti kapacitása, és a zsákkal a maradék három szekció is feloldható majd.

Itt már láthatóak a felvett eszközök a karakterre. Egy felvett fejsze, egy köpeny és két zsák látható felvéve. Az is jól látható, hogy a felvett eszközök kihatással vannak a *CharacterView*-ban látható játékosra, illetve a zsák az *Inventory* kinézetére (azaz a kapacitásra).

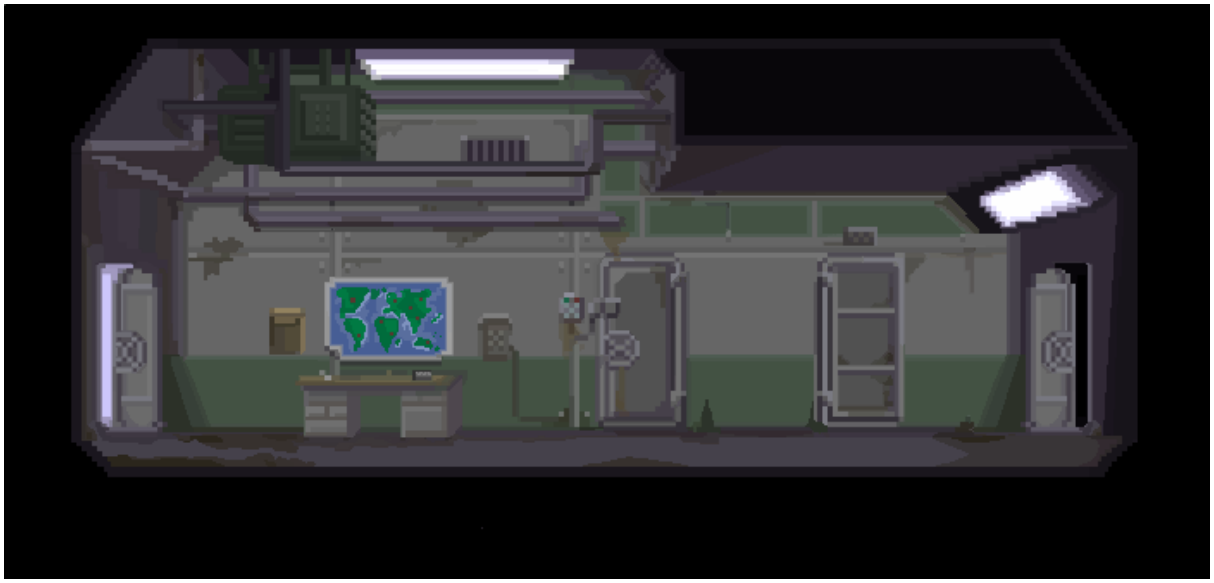
Itt az *Inventory* használata látható már. Jelenleg a játékos rendelkezik egy fejszével egy zsákkal, illetve felvett már 3 db aminósavat is, mely belekerült az *Inventory*-jába. Ha az *Inventory*-ba például eszköz kerül be, akkor annak a felvételét duplaklikkel lehet megvalósítani, illetve ugyanúgy a levétel is így működik. *Inventory*-ból ki is lehet dobni dolgokat, ha már nincs elegendő hely, hiszen a már kicsorbult fejsze nem fog felhalmozódni a már táskában lévő, de még nem használt fejszére, ezért is kell ilyen sok hely.

Így néz ki a kezdetben az úgynevezett *SkillBar*, azaz, ahol az ágensek készítése, tárolása, és az épp karakterre ható ágensek mutatása a felhasználó felé folyik. Bal oldalsó körbe kerül bele az éppen aktuálisan a karakterre ható ágens, lehet az saját maga által feltett ágens, vagy mások által rátett, és lent a bal alsó sarokban jelenik majd meg, hogy hány másodperc maradt hátra még az adott ágens hatóidejéből. Majd [balról jobbra] a belső sávban látható a Vitustánc ikonja, a bénulás, felejtés, végül az immunitás. Itt már két kis rubrika is tartozik egy adott ágenshez, ami arra lesz jó majd, hogyha egy adott ágensből több is elkészült, akkor felső rubrika fogja számlálni a darabszámot, hogy jelenleg hány darab van a játékosnál, alsó rubrikába, meg elkezd visszaszámolni, hogy mennyi ideig lehet még az adott ágenst felhasználni. Mindig a legrégebben elkészült ágens került a „legtetejére a listának”. így ha az elhasználódik, akkor az alatt levő visszaszámlálója váltja fel.

Itt már egy aktív, játék közepén elképzelhető *SkillBar* látható különböző jelzésekkel. Látható, hogy jelenleg a virológusra 1 körig még hat a bénulás, illetve rendelkezik 1 darab Vitustánc támadással, amelyet még 2 körig felhasználhat, 2 darab felejtéssel, melyek közül a legrégebben elkészültet még 3 körig használhatja fel, illetve látszik még, hogy immunitásból szerzett egy genetikai kódot, viszont még nem készítette el az ágenst, melyet a jobb oldalt látható egy gomb, amin egy Erlenymeyer üvegcse van, ha arra rákattintunk, akkor a megpróbál egy ágenst legenerálni, azaz megnézi, hogy van-e megfelelő genetikai kód, illetve megfelelő mennyiségű anyag, és ha van, akkor elkészül az ágens, és hozzá a körös visszaszámláló. Ekkor felkerül annál a gombnál egy 1-es a bal felső sarokban található rubrikába.

11.1.3 Háttérképek

Itt találhatóak a mezők maguk, ugyanis nem „tile game” szerű játékot álmodtunk meg, hanem kihoztuk az egész játékot egy „click and point adventure game”-ként a szebb megjelenítés és a jobb játékélmény érdekében, továbbá nekünk is egyszerűbb volt így elkészíteni, mintsem tile. Első körben itt most csak egy mezőtípust tettünk be példának (sima default/üres mező), majd ugyanazon a mező egy, majd több játékossal (medvetánccal fertőzöttel is), *SkillBar*ral, *Inventory* plusz *CharacterView* meghívásával.



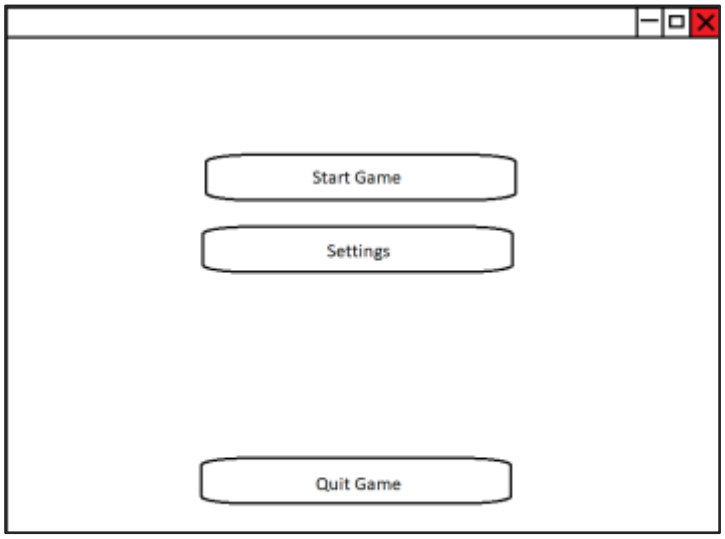
Itt látható az üres háttérkép, amely a háttér lesz az üres mezők során. Még egy kis javítás, további árnyékolások eszközölve lesznek a rendes játékba helyezés előtt.

Játék kezdetében nagyjából ilyen látvány fog tárulni a játékosok elé. Lesz egy üres *SkillBar*, egy üres karakter, tehát nem lesz még rajta semmilyen felszerelés, illetve egy üres *Inventory* is. Ha végzett az adott köri feladatával a *Next*, gombbal tud majd a következő játékosnak átadni a stafétát. Ha az ajtókra kattint, azzal tud majd helyszínt váltani, természetesen az, hogy melyik ajtó mögött mi van, azt nem tudhatja előre. A fényesen fehéren világító ajtó lesz mindig az, ahonnan érkezett. Fent a jobb felső sarokban a kis orvos ikonnal tudja előhívni a karakter-tároló panelt, ami látható a képen is jelenleg, illetve a hang gombbal tudja elnémítani vagy éppen hagyni, hogy a játékban bent lévő zene fusson vagy sem. Az x gomb segítségével tud kilépni a játékból, viszont, ha eközben még van bent más játékos, akkor nekik a játék folytatódik tovább.

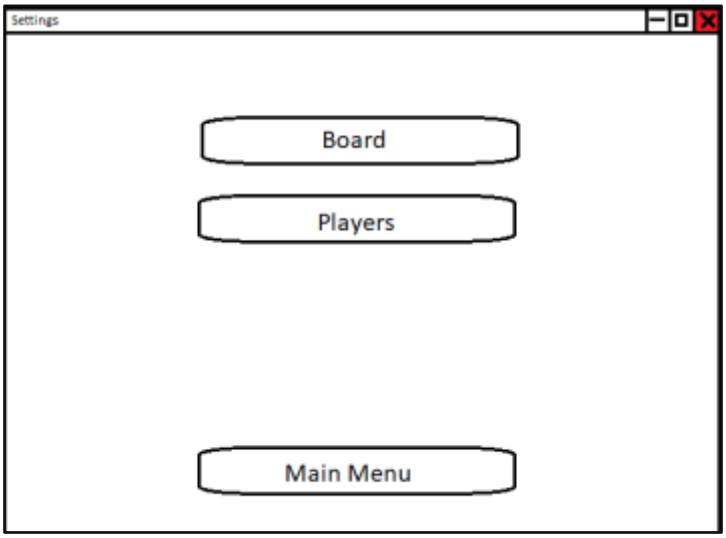
Itt a képen már egy játék közbeni verziót láthatunk. Jelenleg a karakterünk le van bénulva, és a lenti szöveges szekció is szól ezzel kapcsolatban. Ilyenkor nem tudunk rányomni a *Next* gombra, mert automatikusan tovább lépteti a következő játékosra. A képen még látható, hogy van a pályán jelenleg egy medvevírussal fertőzött játékos is, illetve szintén egy, még medvevírussal nem megfertőződött játékos.

11.1.4 Menü

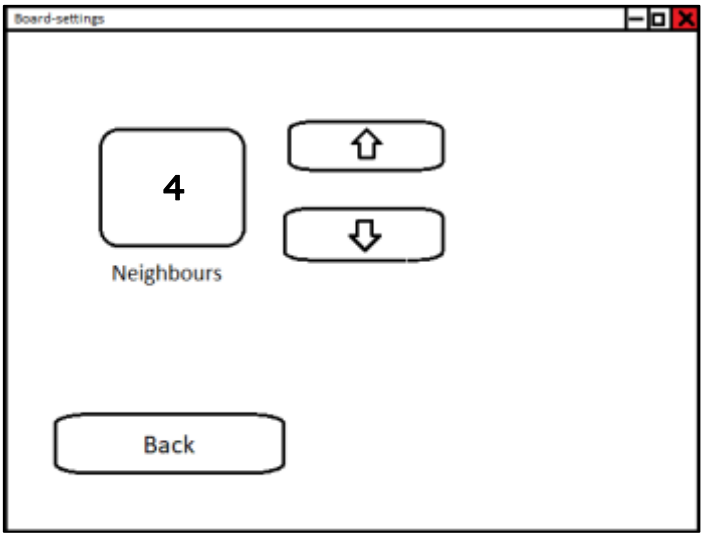
Itt látható egy skicc terv a játék betöltését követő menü rendszerről, és almenüikről.



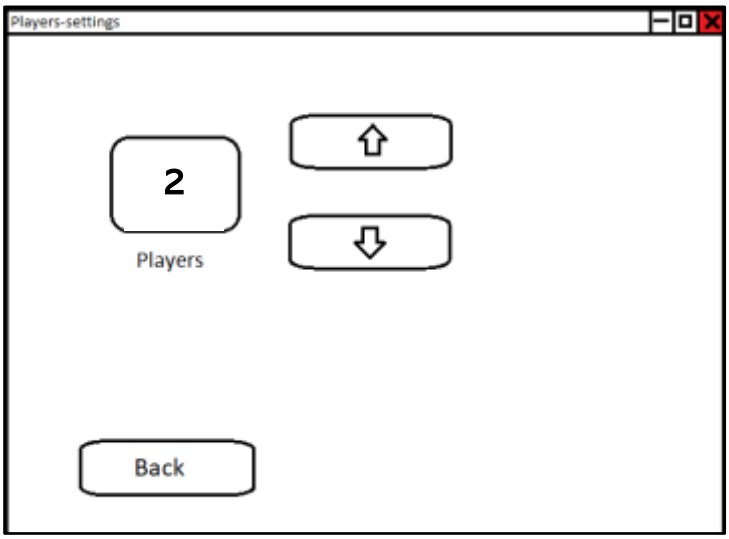
A játék betöltését követően ez a három menüpont vár majd minket, amelyek viszonylag egyértelműen jelzik, hogy mi történik a kattintásukat követően. A *Start Game*-et kiválasztva elkezdődik a játék, és átvált a fentebb látható nézetre.



Abban az esetben, ha a *Setting*sre kattintunk, átmegy egy másik menübe, ahol választhatunk aközül, hogy a pálya kinézetén változtatunk, a játékosok számán, vagy visszamegyünk a főmenübe. Főmenü esetében az előző képet láthatjuk majd.



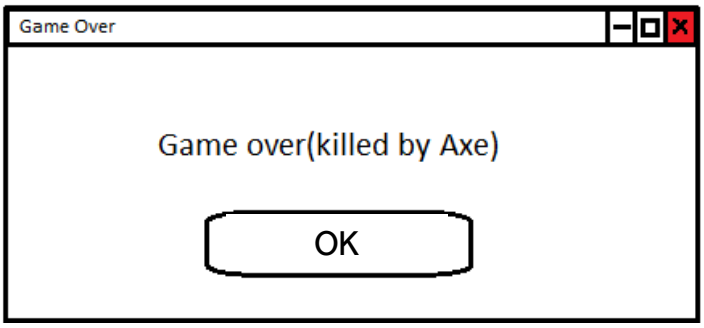
Ha a *Board*ra megyünk, akkor megjelenik ez a nézet, ahol beállíthatjuk, hogy egy mezőnek hány szomszédja lehessen (magyarán milyen alakzatú legyen maga a mező). A default érték 4, a max pedig 6, ezek között nyilakkal lehet mozogni, a bevitt értéket, pedig automatikusan eltárolja. Ha elérte a minimumot, vagy maximumot, akkor a gomb szürkére vált, és nemkattinthatóvá válik. A *Back* gombbal az előző menübe térünk vissza.



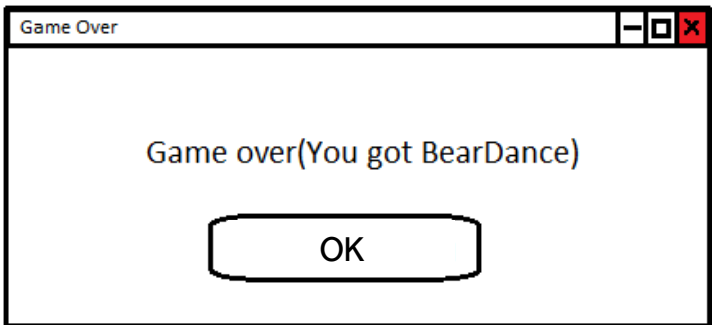
Ha a *Players*re megyünk, akkor megjelenik a már előbbinél látott nézet. Itt meg lehet adni, hogy maximum hány játékos csatlakozhasson be a játékba és ezek alapján fogja a játék indítását követően legenerálni őket. A default érték 2 (tehát legalább egy ellenfél lehessen), a max pedig 4, ezek között nyilakkal lehet mozogni, a bevitt értéket, pedig automatikusan eltárolja. A *Back* gombbal az előző menübe térünk vissza.

11.1.5 Játék vége

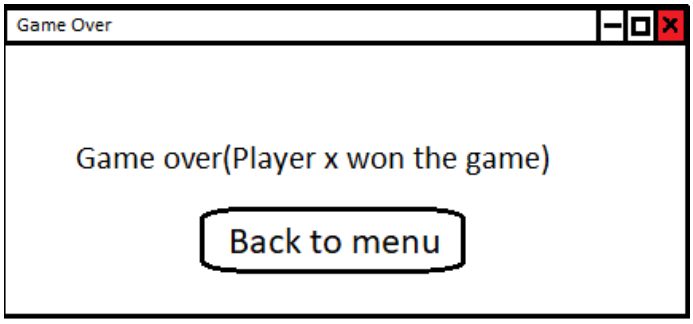
A játék négy különböző kimenetellel végződhet.



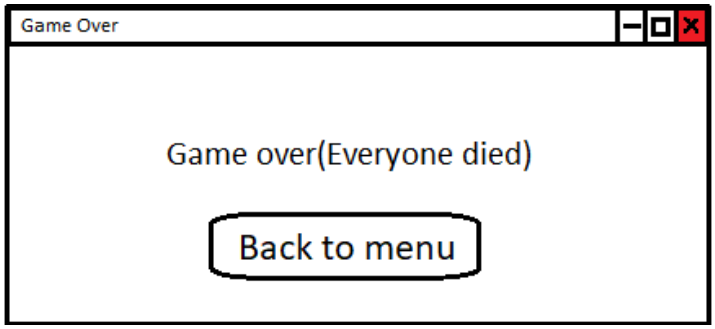
Első eset, amikor a karaktert megölik egy fejszével és úgy ér véget a játék neki. Azonban itt az OK gomb lenyomását követően a többieknek még folytatódik a játék, így automatikusan a soron lévő következő játékost tölti be a játék.



A második eset, hogy medvetánccal fertőződik meg, ebben az esetben az adott játékos már többet nem tud játszani, de az előzővel ellentétben, itt a játékos karaktere még játékban marad, „zombi játékos” lesz belőle.



A harmadik, hogy egy játékos sikeresen összegyűjti az összes genetikai kódot. Ebben az esetben kiíródik a képernyőre, hogy x megnyerte a játékot, így a játék mindenki számára véget ér. Ilyenkor ténylegesen vége a játéknak, és az *OK* gombot felváltja a *Back to menu*, hiszen már nincs mivel játszani.



Ez abban az esetben lép fel, ha mindenki meghal, vagy az utolsó játékos is medvetánccal fertőzött lesz. Itt is hasonlóan az előző verzióban már nem *OK* gomb van.

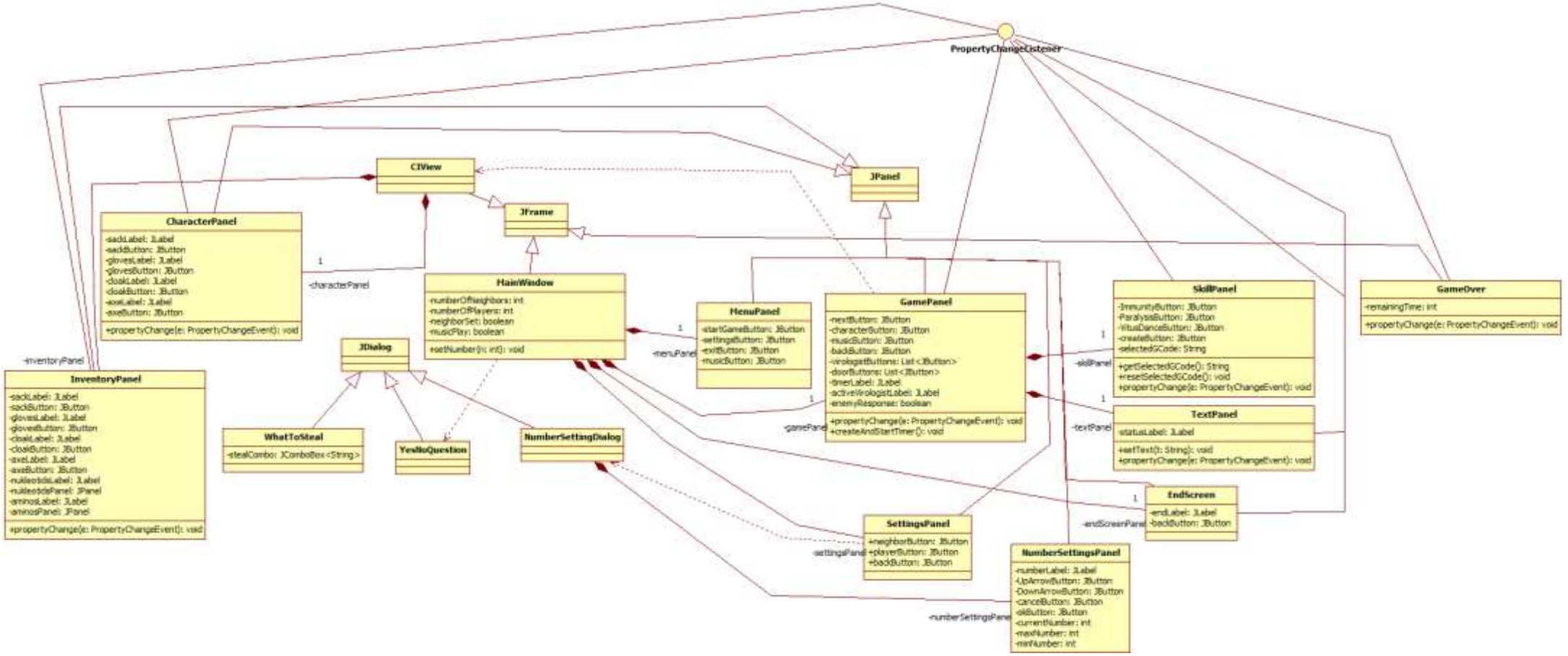
11.2 A grafikus rendszer architektúrája

11.2.1 A felület működési elve

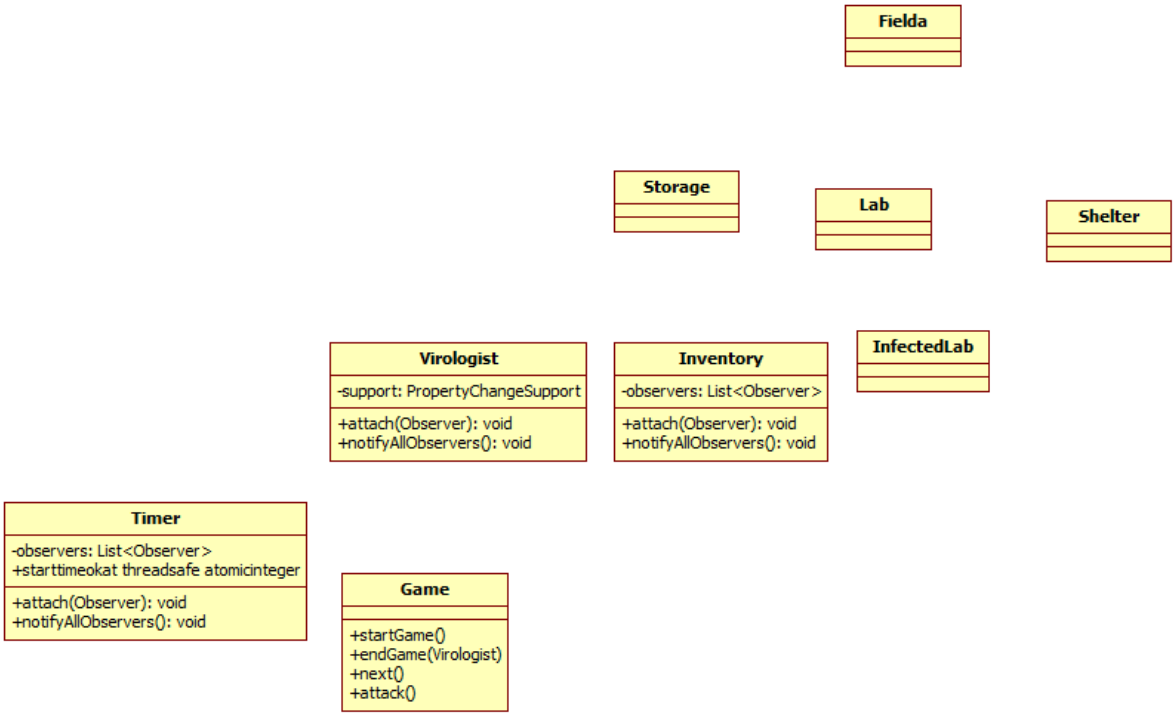
A játék maga egy ablakban fog végig működni a menütől az utolsó percéig a játéknak. Esetlegesen lesznek olyan panelek, amelyek választ, vagy valamilyen fajta reagálást válnak a felhasználótól (pl. mikor megtámadsz valakit, akkor a játék átvált az ő szemszögére, és akkor kap egy ilyen üzenetet, hogy megtámadtak, viszont van nálad egy köpeny, el szeretnéd használni. Ebben az esetben a felhasználónak az igen/nem válaszgombok között kell majd visszareagálnia. Illetve, ha meghívjuk a karakternézőt, akkor az egy előreugró ablakként fog működni, így azt be kell zárni a rajta levő x gomb segítségével). Az egész játékot két módon is bele lehet zárni. A menüben található *Quit Game* segítségével is, illetve az egész játék ablakán található x-gombbal. A játék maga meg van rajzolva, ahogy a háttér is, viszont átlátszó gombokat fogunk helyezni rájuk, amelyek segítik majd a játékost az interakciók tényleges végrehajtásában, így technikailag javarészt mindenhova átlátszó gombokat fogunk elhelyezni a játék szebb megjelenítése érdekében.

11.2.2 A felület osztály struktúrája

11.2.2.1 GUI osztálydiagrammja



11.2.2.2 Eredeti osztálydiagram bővítései



11.3 A grafikus objektumok felsorolása

11.3.1 Game

- **Attribútumok:**
  - **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság***: -, ***Modifier***: static
  - **java.util.Timer roundTimer:** Azért felel, hogy percenként léptesse az aktív játékost. ***Láthatóság***: -, ***Modifier***: static
  - **NextPlayerTask nextPlayerTask:** Ütemezhető taszk, mely run metódusában az idő lejárta után meghívjuk a next parancsot. ***Láthatóság****:* -, ***Modifier****:* static
- **Metódusok:**
  - **void startGame(int neighborCount, int virologistCount)**: A settingsben beállított értékek alapján létrehozzuk a játék mezőit (ügyelve arra, hogy a játék megnyerhető legyen, vagyis minden fajta genetikai kód létezzen valamely labor falára vésve), azokat eltároljuk a Game.fields listában, beállítjuk a szomszédos mezőket, Létrehozzuk a virológusokat (véletlen mezőre állítva őket), azokat hozzáadjuk a Game.virologists listához. A Game.inProgress változót true-ra állítjuk, a roundTimer-re létrehozunk egy új példányt, majd a next parancs meghívásával elindítjuk a játékot. ***Láthatóság:*** *+*, ***Modifier****: static*
    void next(): Az eddigiekben is használt módszerrel beállítjuk a következő játékost. Amennyiben egy játékos cselekvésképtelen, akkor megfelelő üzenet társításával értesítjük róla a gui feliratkozott objektumait. Pl. support.firePropertyChange(„info”, null, „V1 under Paralysis, skip step.”), vagy support.firePropertyChange(„info”, null, „V1 under VitusDance, skip step.”) stb. Amikor találunk egy virológust, aki cselekvésképes, akkor pedig értesítjük róla az erre a változásra feliratkozott gui elemeket a következő metódushívásokkal: support.firePropertyChange(„activeVirologistChange”, null, Game.getActiveVirologist()), illetve a support.firePropertyChange(„fieldChanged”, null, Game.getActiveVirologist().getStandingField()). Korábban ezután a next parancsban busy wait-el dolgoztunk, mely a console-ról várt bemenetet, timeout-al, most azonban mivel már nem a console-ról olvasunk, ez a módszer nem megfelelő. Ehelyett a roundTimer és nextPlayerTask változót használjuk. Ha a nextPlayerTask nem null, akkor meghívjuk annak cancel() metódusát, valamint a roundTimer.purge() metódusát. Ez gondoskodik arról, hogy ha az idő lejárta előtt átadjuk a vezérlést, akkor már ne váltson aktív játékost az előző játékos idejének lejárta után. Ez után új példányt hozunk létre a nextPlayerTask változóba. Meghívjuk az általunk írt Timer singleton osztály startRound() metódusát, és a timer.schedule(nextPlayerTask, 10000) metódushívással újraütemezzük a nextPlayerTask által tárolt objektumot. ***Láthatóság:*** *+*, ***Modifier****: static*
  - **void endGame()**: Az összes virológuson meghívja a die függvényt, a Game inProgress változóját false-ra állítja, meghívja a roundTimer cancel metódusát, majd a support.firePropertyChange(„endGame”, null, v) metódushívással értesíti a rá feliratkozott objektumokat. ***Láthatóság:*** *+*, ***Modifier****: static*
  - **Timer getRoundTimer():** A roundTimer attribútum gettere. ***Láthatóság:*** *+*, ***Modifier****: static*
  - **void setnextPlayerTask(NextPlayerTask newTask):** A nextPlayerTask attribútum settere. ***Láthatóság:*** *+*, ***Modifier****: static*

- **NextPlayerTask getNextPlayerTask():** A nextPlayerTask attribútum gettere. ***Láthatóság:*** *+*, ***Modifier****: static*
- **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** *+*, ***Modifier****: static*
- **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** *+*, ***Modifier****: static*

11.3.2 NextPlayerTask

- **Felelősség:**
  - Adott idő eltelte után meghívja a Game.next() parancsot, majd újraütemezi önmagát
- **Ősosztályok:**
  - java.util.TimerTask
- **Metódusok:**
  - **void run():** Meghívja a Game.next() parancsot.

11.3.3 Virologist

- **Attribútumok:**
- **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság***: -
- **Metódusok:**
- **void equipGloves ()**: Ha bármi miatt meghiúsul a kesztyű felvétele (pl. túl sok eszköz van rajtunk), akkor support.firePropertyChange(„error”, null, errorMessage) metódushívással értesítjük a rá feliratkozott gui objektumokat. Egyébként, ha az inventoryból kivett kesztyű nem null, akkor a virológus wornGloves listájához való hozzáadás után support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság****: +*
- **void equipSack ()**: Ha bármi miatt meghiúsul a zsák felvétele (pl. túl sok eszköz van rajtunk), akkor support.firePropertyChange(„error”, null, errorMessage) metódushívással értesítjük a rá feliratkozott gui objektumokat. Egyébként, ha az inventoryból kivett zsák nem null, akkor a virológus wornSack listájához való hozzáadás, és az inventory maxMaterial növelése után
  - support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság****: +*
- **void equipCloak():**Ha bármi miatt meghiúsul a köpeny felvétele (pl. túl sok eszköz van rajtunk), akkor support.firePropertyChange(„error”, null, errorMessage) metódushívással értesítjük a rá feliratkozott gui objektumokat. Egyébként, ha az inventoryból kivett köpeny nem null, akkor a virológus wornCloak listájához való hozzáadás után support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság****: +*
- **void equipAxe():**Ha bármi miatt meghiúsul a fejsze felvétele (pl. túl sok eszköz van rajtunk), akkor support.firePropertyChange(„error”, null, errorMessage) metódushívással értesítjük a rá feliratkozott gui objektumokat. Egyébként, ha az

inventoryból kivett fejsze nem null, akkor a virológus wornAxe listájához való hozzáadás után support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság**: +*

- **void takeOffGloves():** Ha volt eszköz, amit levehettünk, vagyis az if(wornGloves.size()>0) részen belül, minden művelet elvégzése után a support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság**: +*
- **void takeOffSack(): Bugfix**: Ha a zsák nem az inventory-ba kerül, de le tudjuk venni magunkról, akkor is csökkenteni kell az inventory.maxMaterial értékét. **Egyéb módosítás**: Ha volt eszköz, amit levehettünk, vagyis az if(wornSack.size()>0) részen belül, minden művelet elvégzése után a support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság**: +*
- **void takeOffCloak():**Ha volt eszköz, amit levehettünk, vagyis az if(wornCloak.size()>0) részen belül, minden művelet elvégzése után a support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság**: +*
- **void takeOffAxe():**Ha volt eszköz, amit levehettünk, vagyis az if(wornAxe.size()>0) részen belül, minden művelet elvégzése után a
    - support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság**: +*
    - **boolean assaulted(Virologist vTo, Agent a):** Ha a virológus valamilyen ágens hatása alatt van, akkor support.firePropertyChange(„error”, null, errorMessage) metódushívással értesítjük a rá feliratkozott gui objektumokat. Ha nem sikerült a védekezés, akkor a support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság**: -*
- **void createAgent(GCode gc):** Ha az ágenst sikeresen hozzáadtuk a felhasználható ágensek listájába, akkor support.firePropertyChange(„skills”, null, agents) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság**: +*
- **void setInventory():** *miután beállítottuk az inventory-t,* support.firePropertyChange(„inventory”, null, getInventory()) metódushívással értesítjük a gui feliratkozott példányait. ***Láthatóság**: +*
- **void die():** *miután elvégeztük a szükséges műveleteket,* support.firePropertyChange(„info”, null,getID() + „ died.”) metódushívással értesítjük a gui feliratkozott példányait. ***Láthatóság**: +*
- **boolean wannaDrop()**: Már nem console-on olvassuk be azt, hogy a felhasználó az inventory-ba szeretne-e tenni egy eszközt vagy eldobni, hanem egy dialogwindow-ot jelenítünk meg rá, melynek time limit-et állítunk be. A time limit-nek a
    - Timer.getRemainingTime() metódus által visszaadott értéket adjuk meg. Ha az idő lejár, a DialogWindow-ot az X gombbal bezárjuk, vagy a No gombra kattintunk, akkor a válasz nem, és a metódus false-al tér vissza. Ha a Yes gombra kattintunk, akkor a válasz igen, és a metódus true-val tér vissza. ***Láthatóság**: +*
- **boolean wannaSteal(Virologist vTo)**: Ez a metódus egy duplikáció volt, töröljük, és mindenhol, ahol ezt hívtuk, a virológus standingField-jének a wannaSteal metódusát hívjuk helyette.
- **void move(Field f, boolean bEmittedByUser):** Miután a virológust eltávolítottuk a régi mezőjéről, majd hozzáadtuk az új mezőhöz, de még az arrive meghívása előtt a support.firePropertyChange(„fieldChanged”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság**: +*

- **String chooseObjectToSteal():** Már nem console-on olvassuk be azt, hogy a felhasználó mit szeretne lopni az adott virológustól, hanem egy dialogwindow-ot jelenítünk meg rá, melynek time limit-et állítunk be. A time limit-nek a Timer.getRemainingTime() metódus által visszaadott értéket adjuk meg. A
  - DialogWindow-on van egy JComboBox, melyben ki lehet választani az ellopni kívánt Equipment-et vagy Material-t. Ha a DialogWindow-ot az X gombbal bezárjuk, akkor a metódus üres string-el tér vissza. Ha az idő lejár, vagy a DialogWindow-on az ok gombra kattintunk, akkor a ComboBox-ban kiválasztott objektumnévvel tér vissza a metódus. ***Láthatóság:*** *-*
- **boolean defense(Virologist vFr, Agent a):** Ha eltávolítunk egy viselt kesztyűt, vagy köpenyt, akkor a support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. Ha a Game.getActiveVirologist() nem egyenlő a this által mutatott objektummal, akkor az Axe.wannaUse és a Gloves.wannaUse előtt meghívjuk a support.firePropertyChange(„activeVirologistChange”, null, this) metódust, amivel azt jelezzük, hogy ennél a DialogWindow-nál másik játékos kell, hogy válaszoljon. ***Láthatóság:*** *+*
- **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** *+*
- **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** *+*

11.3.4 Timer

- **Attribútumok:**

- **java.util.concurrent.atomic.AtomicLong startTime**: az az időpont, amikor az aktív virológus sorra került ***Láthatóság:*** *-*, ***Modifier****: static*
- **ava.util.concurrent.atomic.AtomicLong enemyStartTime**: az az időpont, amikor átadtuk a vezérlést rövidebb időre egy másik játékosnak

- **Metódusok:**

- **void startRound()**: A funkcionalitás nem változik, csak átalakítjuk a módosított atomic változó használatához. ***Láthatóság:*** *+*, ***Modifier****: static*
- **void startEnemyRound()**: A funkcionalitás nem változik, csak átalakítjuk a módosított atomic változó használatához. ***Láthatóság:*** *+*, ***Modifier****: static*
- **long getEnemyRemainingTime()**: A funkcionalitás nem változik, csak átalakítjuk a módosított atomic változó használatához. ***Láthatóság:*** *+*, ***Modifier****: static*
- **long timeSpentByEnemy():** A funkcionalitás nem változik, csak átalakítjuk a módosított atomic változó használatához. ***Láthatóság:*** *+*, ***Modifier****: static*
- **long getRemainingTime():**A funkcionalitás nem változik, csak átalakítjuk a módosított atomic változó használatához. ***Láthatóság:*** *+*, ***Modifier****: static*
- **void increaseRemainingTime(long time)**: Nem csak a remainingTime-ot növeljük meg a bemeneti paraméter értékével, hanem elkérjük a Game nextPlayerTask attribútumát, és meghívjuk annak cancel() metódusát. Ez után a Game roundTimer objektumát is elkérjük, és meghívjuk annak purge() metódusát. Végül új példányt hozunk létre a nextPlayerTask változóba, és a timer.schedule(nextPlayerTask, getRemainingTime()) metódushívással újraütemezzük a nextPlayerTask által tárolt objektumot. ***Láthatóság:*** *+*, ***Modifier****: static*

11.3.5 Inventory

- **Attribútumok:**
  - **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. **Láthatóság**: -
- **Metódusok:**
  - **void setMaxMaterial(int maxMaterial)**: Felveszünk két új belső változót, melyekben eltároljuk, hogy mennyi aminoacid, illetve mennyi nukleotid került eldobásra, és ha bármelyik nagyobb, mint 0, akkor értesítjük az ezen osztály változásaira feliratkozott gui elemeket a support.firePropertyChange(„info”, null, msgInfo), illetve a support.firePropertyChange(„inventory”, null, this) metódushívásokkal. ***Láthatóság:*** +
  - **void addGloves(Gloves g):** Ha nem sikerül a kesztyű hozzáadása az inventory-hoz, akkor support.firePropertyChange(„error”, null, errInfo) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket, ha sikerül, akkor pedig a support.firePropertyChange(„inventory”, null, this) metódust hívjuk. ***Láthatóság:*** + **void addSack(Sack s):** Ha nem sikerül a zsák hozzáadása az inventory-hoz, akkor support.firePropertyChange(„error”, null, errInfo) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket, ha sikerül, akkor pedig a support.firePropertyChange(„inventory”, null, this) metódust hívjuk. ***Láthatóság:*** +
  - **void addCloak(Cloak c):** Ha nem sikerül a köpeny hozzáadása az inventory-hoz, akkor support.firePropertyChange(„error”, null, errInfo) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket, ha sikerül, akkor pedig a support.firePropertyChange(„inventory”, null, this) metódust hívjuk. ***Láthatóság:*** +
  - **void addAxe(Axe a):** Ha nem sikerül a fejsze hozzáadása az inventory-hoz, akkor support.firePropertyChange(„error”, null, errInfo) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket, ha sikerül, akkor pedig a support.firePropertyChange(„inventory”, null, this) metódust hívjuk. ***Láthatóság:*** +
  - **Gloves removeGloves():** Ha sikerül eltávolítani a kesztyűt, akkor a support.firePropertyChange(„inventory”, null, this) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket. . ***Láthatóság:*** +
  - **Sack removeSack():**Ha sikerül eltávolítani a zsákot, akkor a support.firePropertyChange(„inventory”, null, this) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket. . ***Láthatóság:*** +
  - **Cloak removeCloak()**: Ha sikerül eltávolítani a köpenyt, akkor a support.firePropertyChange(„inventory”, null, this) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket. . ***Láthatóság:*** +
  - **Axe removeAxe():** Ha sikerül eltávolítani a fejszét, akkor a support.firePropertyChange(„inventory”, null, this) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket. . ***Láthatóság:*** +
  - **int learnGCode(GCode g):** Ha új kódot tanultunk, akkor support.firePropertyChange(„inventory”, null, this) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket. . ***Láthatóság:*** +
  - **void forgetGCodes():** Az ismert genetikai kódok listájának kiürítése után support.firePropertyChange(„inventory”, null, this) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket. . ***Láthatóság:*** +
  - **void addAmino(Aminoacid am)**: Ha nem sikerül az Aminoacid hozzáadása az inventory-hoz, akkor support.firePropertyChange(„error”, null, errInfo)

metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket, ha sikerül, akkor pedig a support.firePropertyChange(„inventory”, null, this) metódust hívjuk. ***Láthatóság:*** +

- **void addNukleo(Nukleotid n)**: Ha nem sikerül a Nukleotid hozzáadása az inventoryhoz, akkor support.firePropertyChange(„error”, null, errInfo) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket, ha sikerül, akkor pedig a support.firePropertyChange(„inventory”, null, this) metódust hívjuk. ***Láthatóság:*** +
- **Aminoacid removeAmino():** Ha sikerül eltávolítani az Aminoacidot, akkor a support.firePropertyChange(„inventory”, null, this) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket. . ***Láthatóság:*** +
- **Nukleotid removeNukleo():** Ha sikerül eltávolítani a Nukleotidot, akkor a support.firePropertyChange(„inventory”, null, this) metódushívással értesítjük az ezen osztály változásaira feliratkozott gui elemeket. . ***Láthatóság:*** +
- **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** +
- **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** +

11.3.6 BearDance

- **Attribútumok:**

- **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság:*** -

- **Metódusok:**

- **void setVirologistUnderEffect(Virologist virologist)**: A metódus végén meghívjuk a support.firePropertyChange(„virologist”, null, virologist) -t, illetve a support.firePropertyChange(„gameOverByBearDance”, null, virologist) -t. ***Láthatóság:*** +
- **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** +
- **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** +

11.3.7 Immunity

- **Attribútumok:**
  - **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság:*** -
- **Metódusok:**
  - **void setVirologistUnderEffect(Virologist virologist)**: A metódus végén meghívjuk a support.firePropertyChange(„virologist”, null, virologist) -t. ***Láthatóság:*** +
  - **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** +
  - **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** +

11.3.8 Oblivion

- **Attribútumok:**
  - **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság:*** -
- **Metódusok:**
  - **void setVirologistUnderEffect(Virologist virologist):** A metódus végén meghívjuk a support.firePropertyChange(„virologist”, null, virologist) -t. ***Láthatóság:*** +
  - **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** +
  - **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** +

11.3.9 Paralysis

- **Attribútumok:**
  - **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság:*** -
- **Metódusok:**
  - **void setVirologistUnderEffect(Virologist virologist)**: A metódus végén meghívjuk a support.firePropertyChange(„virologist”, null, virologist) -t. ***Láthatóság:*** +
  - **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** +
  - **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** +

11.3.10 Gloves

- **Metódusok:**
  - **boolean wannaUse():** Már nem console-on olvassuk be azt, hogy a felhasználó akarja-e használni a kesztyűt, hanem egy dialogwindow-ot jelenítünk meg rá, melynek time limit-et állítunk be. A time limit-nek a Timer.getRemainingTime() metódus által visszaadott értéket adjuk meg. Ha az idő lejár, a DialogWindow-ot az X gombbal bezárjuk, vagy a No gombra kattintunk, akkor a válasz nem, és a metódus false-al tér vissza. Ha a Yes gombra kattintunk, akkor a válasz igen, és a metódus true-val tér vissza. ***Láthatóság:*** +

11.3.11 Axe

- **Attribútumok:**
  - **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság:*** -
- **Metódusok:**
  - **boolean wannaUse():** Már nem console-on olvassuk be azt, hogy a felhasználó akarja-e használni a fejszét, hanem egy dialogwindow-ot jelenítünk meg rá, melynek time limit-et állítunk be. A time limit-nek a Timer.getRemainingTime() metódus által visszaadott értéket adjuk meg. Ha az idő lejár, a DialogWindow-ot az X gombbal bezárjuk, vagy a No gombra kattintunk, akkor a válasz nem, és a metódus false-al tér vissza. Ha a Yes gombra kattintunk, akkor a válasz igen, és a metódus true-val tér vissza. ***Láthatóság:*** +
  - **void use(Virologist v):** A metódus végén meghívjuk a support.firePropertyChange(„virologist”, null, v) -t. ***Láthatóság:*** +
  - **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** +
  - **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** +

11.3.12 GCode

- **Attribútumok:**
  - **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság:*** -
- **Metódusok:**
  - **public Agent create (Virologist v):** Ha az ágens létrehozása nyersanyaghiány (Material) miatt meghiúsul, akkor support.firePropertyChange(„error”, null, errorMessage) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság:*** +
  - **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** +

- **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** +

11.3.13 Field

- **Metódusok:**
  - **boolean wannaSteal(Virologist v):** Már nem console-on olvassuk be azt, hogy a felhasználó akar-e lopni, hanem egy dialogwindow-ot jelenítünk meg rá, melynek time limit-et állítunk be. A time limit-nek a Timer.getRemainingTime() metódus által visszaadott értéket adjuk meg. Ha az idő lejár, a DialogWindow-ot az X gombbal bezárjuk, vagy a No gombra kattintunk, akkor a válasz nem, és a metódus false-al tér vissza. Ha a Yes gombra kattintunk, akkor a válasz igen, és a metódus true-val tér vissza. ***Láthatóság:*** +

11.3.14 Lab

- **Metódusok:**
  - **boolean wannaTouchy():** Már nem console-on olvassuk be azt, hogy a felhasználó akar-e kódot letapogatni, hanem egy dialogwindow-ot jelenítünk meg rá, melynek time limit-et állítunk be. A time limit-nek a Timer.getRemainingTime() metódus által visszaadott értéket adjuk meg. Ha az idő lejár, a DialogWindow-ot az X gombbal bezárjuk, vagy a No gombra kattintunk, akkor a válasz nem, és a metódus false-al tér vissza. Ha a Yes gombra kattintunk, akkor a válasz igen, és a metódus true-val tér vissza. ***Láthatóság:*** +

11.3.15 InfectedLab

- **Attribútumok:**
  - **PropertyChangeSupport support**: az osztályon belül történt változások esetén értesítést küld a grafikus interface elemeinek, melyeknek frissíteniük kell a megjelenített adatait. ***Láthatóság:*** -

- **Metódusok:**
  - **boolean arrive(Virologist v):** Ha a virológusról eltávolítunk egy köpenyt, akkor a support.firePropertyChange(„virologist”, null, this) metódushívással értesítjük a rá feliratkozott gui objektumokat. ***Láthatóság:*** +
  - **void addPropertyChangeListener(PropertyChangeListener pcl)**: feliratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követésére. ***Láthatóság:*** +
  - **void removePropertyChangeListener(PropertyChangeListener pcl)**: leiratkoztat egy PropertyChangeListener interface-t implementáló objektum példányát az osztályban történt módosítások követéséről. ***Láthatóság:*** +

11.3.16 Storage

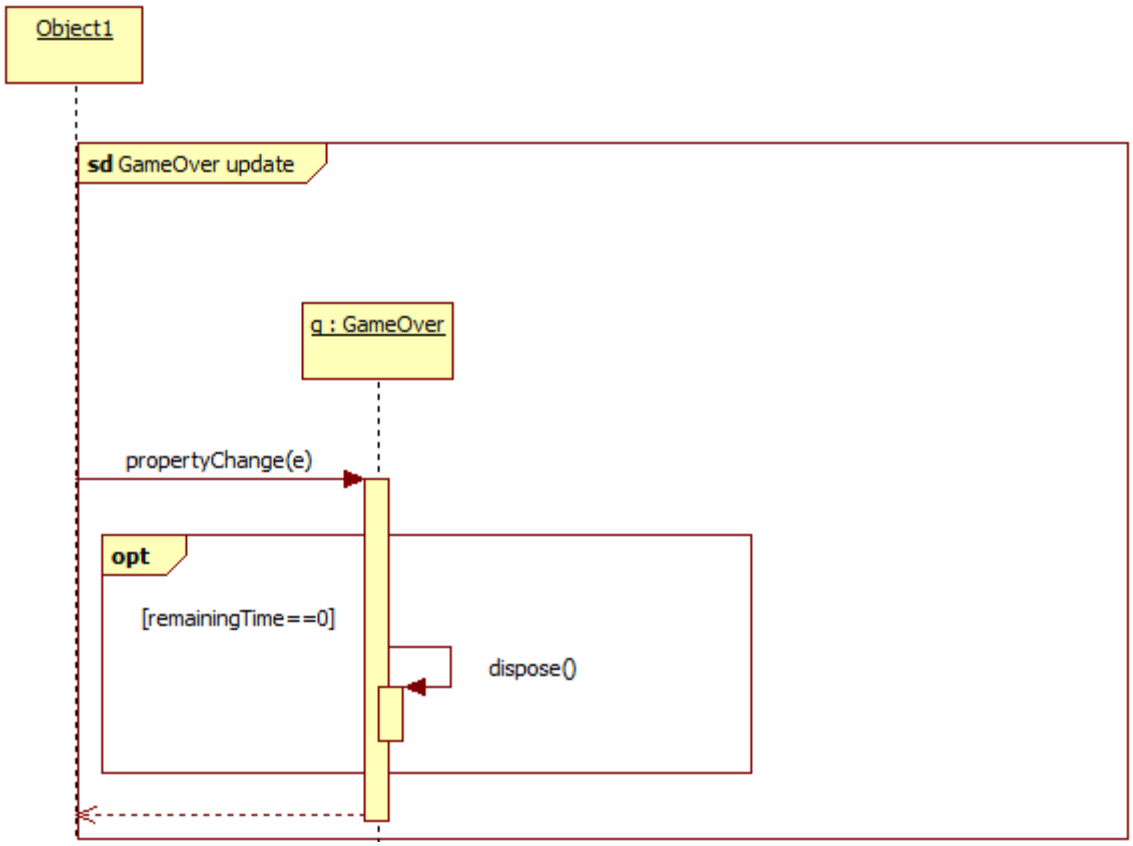
- **Metódusok:**
  - **boolean wannaPickup():** Már nem console-on olvassuk be azt, hogy a felhasználó akar-e nyersanyagot (Material) felvenni, hanem egy dialogwindow-ot jelenítünk meg rá, melynek time limit-et állítunk be. A time limit-nek a Timer.getRemainingTime() metódus által visszaadott értéket adjuk meg. Ha az idő lejár, a DialogWindow-ot az X gombbal bezárjuk, vagy a No gombra kattintunk, akkor a válasz nem, és a metódus false-al tér vissza. Ha a Yes gombra kattintunk, akkor a válasz igen, és a metódus trueval tér vissza. ***Láthatóság**: +*

11.3.17 Shelter

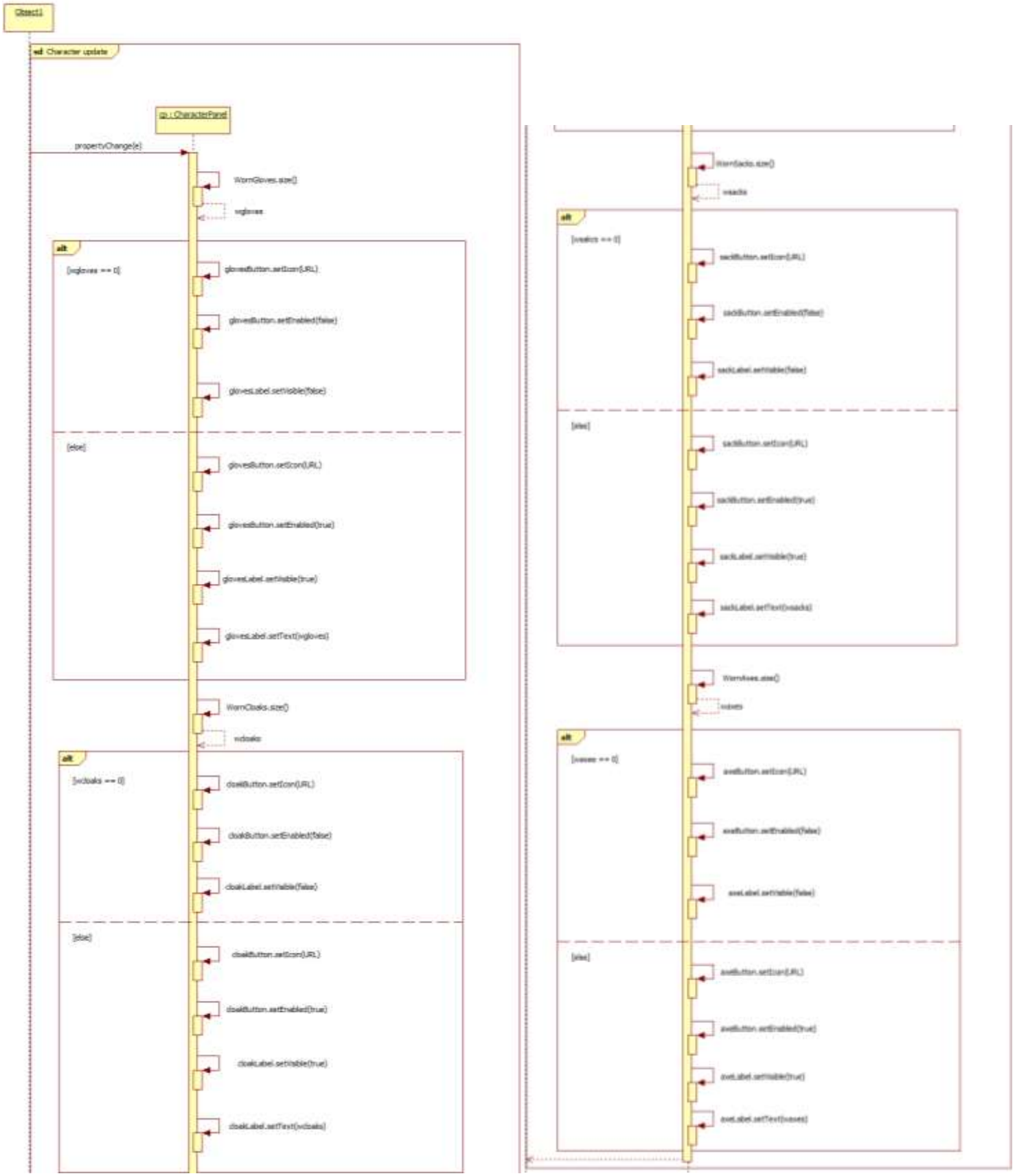
- **Metódusok:**
  - **boolean wannaPickup():** Már nem console-on olvassuk be azt, hogy a felhasználó akar-e eszközt (Equipment) felvenni, hanem egy dialogwindow-ot jelenítünk meg rá, melynek time limit-et állítunk be. A time limit-nek a Timer.getRemainingTime() metódus által visszaadott értéket adjuk meg. Ha az idő lejár, a DialogWindow-ot az X gombbal bezárjuk, vagy a No gombra kattintunk, akkor a válasz nem, és a metódus false-al tér vissza. Ha a Yes gombra kattintunk, akkor a válasz igen, és a metódus trueval tér vissza. ***Láthatóság**: +*

11.4 Kapcsolat az alkalmazói rendszerrel

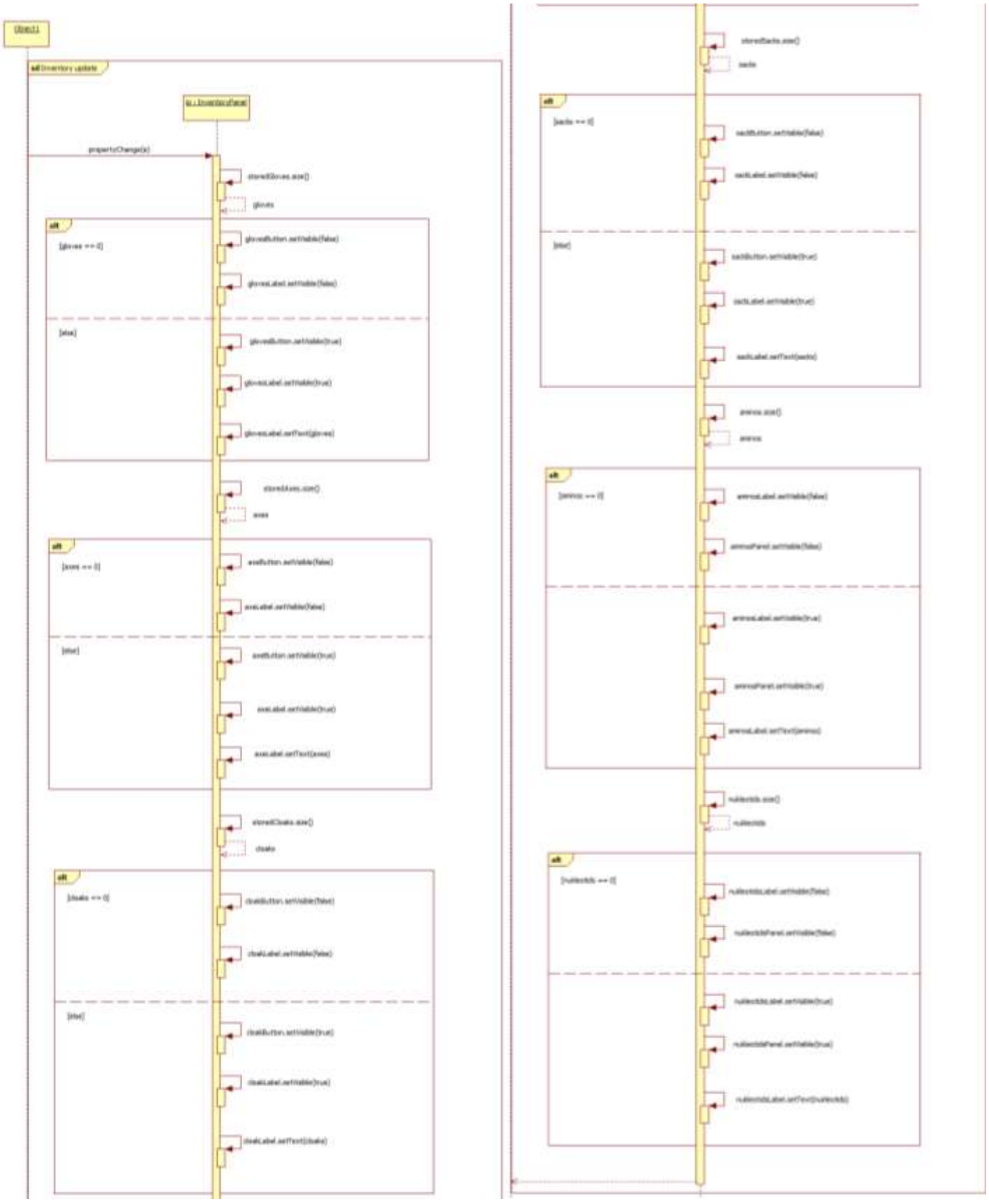
11.4.1 GameOver



11.4.2 Character



11.4.3 Inventory



11.5 Ütemterv

Határidő	Feladat	Pont	Felelős
febr. 28.	Követelmény, projekt, funkcionalitás - beadás	10	Csia
márc. 7.	Analízis modell kidolgozása I. - beadás	20	Alpek
márc. 16.	Analízis modell kidolgozása II. - beadás	30	Marton
márc. 21.	Szkeleton tervezése - beadás	20	Litavecz
márc. 28.	Szkeleton elkészítése - beadás	20	Csia
ápr. 4.	Prototípus koncepciója – beadás Szkeleton bemutatás	20	Alpek
ápr. 11.	Részletes tervek - beadás	45	Marton
ápr. 25.	Prototípus készítése, tesztelése	35	Litavecz
máj. 2.	Grafikus változat tervei – beadás Prototípus - beadás és a forráskód, a tesztbemenetek és az elvárt kimenetek herculesre való feltöltése	30	Ruskó
máj. 16.	Grafikus változat készítése	40	
máj. 18.	Egységes dokumentáció - beadás és bemutatás Grafikus változat - beadás és a forráskód herculesre való feltöltése, és teljes házi bemutatás	30	

11.6 Napló

Kezdet	Időtartam	Résztvevő(k)	Leírás
2022.04.11 17:40	1 óra	Alpek, Csia, Ruskó	Megbeszélés: Grafikus rész alapötletek megvitatása.
2022.04.11 – 05.02	Kb. 30 óra	Csia	Feladat: Elemek, hátterek, karakterek, objektumok egyedileg megtervezése, Pixelart nevű program segítségével megrajzolásuk.
2022.05.01 11:20	1,5 óra	Csia, Ruskó	Megbeszélés: Osztálydiagram alapvető átbeszélése, alapvető ötletek egyeztetése a grafikai kinézettel kapcsolatban.
2022.05.01 13:40	2 óra 20 perc	Csia, Ruskó	Megbeszélés: További ötletek, problémák megvitatása a grafikus kinézettel kapcsolatban, osztálydiagram és állapotgépek megrajzolása a megbeszéltek szerint.
2022.05.01 16:00	3 óra	Ruskó	Feladat: Osztálydiagram leírások elkészítése.
2022.05.01 17:20	1,5 óra	Csia	Feladat: Grafikus interfész szekció összeállítása, szövegezések megírása.
2022.05.02. 17:00	6 óra	Alpek, Csia, Marton, Ruskó	Megbeszélés: Osztálydiagram teljes átszervezése, feladatok szétosztása.
2022.05.03 06:00	1 óra	Alpek	Feladat: Menü és játék vége skiccek elkészítése.
2022.0503 11:00	2 óra	Csia	Feladat: Menü és játék vége részekhez szövegezés elkészítése, A felület működési elv rész megírása.
2022.05.03 15:00	8 óra	Marton	Feladat: Osztálydiagrammok leírásainak elkészítése.
2022.05.04 01:30	3 óra	Litavecz	Feladat: Szekvenciadiagrammok megrajzolása.
2022.05.04 09:00	40 perc	Csia	Feladat: Dokumentum véglegesítése összegzése

11.7 Százalékos teljesítés

Név	Százalék
Alpek	20%
Csia	20%
Litavecz	20%
Marton	20%
Ruskó	20%