Szakdolgozat

Miskolci Egyetem

Multiágens rendszerek koordinációs problémáinak vizsgálata

Készítette:

Mengyán Márton

Programtervező informatikus

Témavezető:

Piller Imre

Miskolc, 2022

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

Szakdolgozat Feladat

Mengyán Márton (BPWTW0) programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: szimuláció, ágens

A szakdolgozat címe: Multiágens rendszerek koordinációs problémáinak vizsgálata

A feladat részletezése:

A multiágens rendszerek célja, hogy egy adott problémát egyidejűleg több ágens irányításával tudjanak megoldani.
A dolgozat azt vizsgálja, hogy ilyen esetekben az ágensek között milyen jellegű kommunikáció lehet, illetve hogy az adott ágensnek hogyan kell viselkednie a környezetből, és más ágensektől érkező információk függvényében.
Ehhez specifikálásra, megtervezésre és implementálásra kerül egy szimulációs környezet, amely valós időben követhetővé teszi az ágensek állapotát.
Az ágensek viselkedéséhez tartozó paraméterek optimalizálásra kerülnek annak érdekében, hogy az adott célt minél hatékonyabban (például gyorsabban vagy kevesebb művelet elvégzésével) el tudják érni.

Témavezető: Piller Imre (egyetemi tanársegéd)

A feladat kiadásának ideje: 2021. Szeptember 27.

. . . . . .
szakfelelős

Eredetiségi Nyilatkozat

Alulírott **Mengyán Márton**; Neptun-kód: `BPWTW0` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Multiágens rendszerek koordinációs problémáinak vizsgálata* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ...év ..... .hó ..... .nap

.....

Hallgató

1.

szükséges (módosítás külön lapon)

A szakdolgozat feladat módosítása

nem szükséges

....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

....

....

....

3. A szakdolgozat beadható:

....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

....

dátum témavezető(k)

5.

bocsátható

A szakdolgozat bírálatra

nem bocsátható

A bíráló neve: .....

....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ...

a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. fejezet

Bevezetés

A szakdolgozat egy szimulációs környezet létrehozásával fog kezdődni, amelyet felül nézetből vehetünk szemügyre. Körökre osztott, az ágenseket egy fix sorrendbe rakjuk, amely meghatározza, mikor jön melyik ágens. A szimulációs környezet véges számú különböző szobákból és folyósókból épül fel. Két szobát egy folyosó köt össze, amelyet a folyosó mindkét végén egy ajtó zár le.

Az ágensek két nagyobb csoportra vannak osztva, amelyeket 'frakció'-knak nevezünk. Minden frakciónak van egy teljesítendő célja, amely teljesítésénél véget ér a szimuláció. Az ágensek miközben keresik a célukat, figyelembe veszik az útközben észrevett tárgyakat, kulcsokat, ajtókat és ellenséges ágenseket és ezek alapján tervezik meg következő lépésüket.

A szobák és folyosók blokkokból épülnek fel. Ezeken a blokkokon helyezkedhetnek el a karakterek limitáltan, azaz egynél több karakter nem tartózkodhat rajta.

A program Java programozási nyelven [1] készül, a *Java Swing* csomag [3] használatával.

2. fejezet

Koncepció

2.1. A fejezet célja

A szimulációs környezet inspirációt vesz már elkészített videójátékokból, mint például a *Pixel Dungeon* [5] és *Darkest Dungeon* [4] nevű játékokból ismerhető szoba és folyosó kapcsolat. Ahol minden szobát egy folyosó köt össze egy másik tetszőleges szobával. A Darkest Dungeon-hoz képest a játékmenet rugalmasabb, mivel a szobák bármikor elhagyhatóak és megközelíthetőek, ha engedi azt a pálya jelenlegi belső felépítése. Azaz, ha nincs útban valami, ami megakadályozza.

A fent említett két játék mind körökre osztott, ahogyan a szimulációs környezetünk is az lesz.

A szimulációs környezetünk viszont egyedi lesz abból a szempontból, hogy az ágensek csoportokra vannak osztva és közös céllal rendelkezve próbálják legyőzni egymást és az ellenséges ágenseket.

Az alkalmazás a *Pela* fantázia nevet kapta.

2.2. Felhasznált technológiák

2.2.1. Java

A Java általános célú, objektumorientált programozási nyelv, amelyet a Sun Microsystems fejlesztett a 90-es évek elejétől kezdve egészen 2009-ig, amikor a céget felvásárolta az Oracle [1].

2.2.2. Java Swing

A Swing osztályok kiküszöbölik a Java legnagyobb gyengeségét, a viszonylag primitív felhasználói felület eszközkészletét. A Java Swing segít a Swing osztályok teljes előnyeinek kihasználásában, részletes leírást adva minden osztályról és felületről a kulcsfontosságú Swing csomagokban [3].

2.3. Az ágens szimulációs felület elemei

Az ágensek és környezetük szimulációjához, a rajtuk elvégzendő mérésekhez célszerű összerakni egy grafikus felületet. A következő szakaszokban ezeknek a két fő elemét, a

menüt és az UI-t mutatja be a dolgozat.

2.3.1. Menü

A szimulációs környezethez tartozik egy menü ablak. Ennek elemei az alábbiak.

Szimuláció indítása előtt:

- Középen felül a *Main Menu* kiírását láhatjuk, alatta pedig 2 opcióból választhatunk (2.5. ábra):
  (1) *Start*, a kiválasztása esetén elindul a szimuláció,
  (2) *Quit*, a szimulációs ablak bezárása.
- A fenn említett két menüpont között a `W` (fel) és `S` (le) billentyűk lenyomásával navigálhatunk.
- Aláhúzva jelzi a felhasználónak, hogy mely menüpont van éppen kiválasztva.
- A különböző ablakok és UI elemek egy egyedi fontot használnak [2].

A szimuláció megállítása esetén:

- A szimulációban a `P` billentyű lenyomásával képes a felhasználó megállítani a szimuláció menetét.
- A megállítást a bal felső sarokban kiírt fehér *Paused* kiírás jelzi a felhasználó számára (2.2. ábra).
- Ilyenkor az ágensek nem kapnak lehetőséget a lépésre, de a felhasználó képes a kamera mozgatására a nyilak segítségével.
- A felhasználó szintén képes az ágensek *Statistics* ablakát megnyitni/bezárni a `C` billentyű megnyomásával, és előre/hátra lapozni az ágensek között a `Q` (balra) és `E` (jobbra) billentyűk megnyomásával, ha több van a szimulációban, mint 1.
- Ugyanígy képes megnyitni/bezárni az adott ágenshez tartozó *Inventory* ablakot az `I` billentyű megnyomásával, ha látszódik az ágens statisztikája ablak.
- A `P` billentyű újonnan megnyomásával ott folytatódik a szimuláció, ahol abbamaradt (2.1. ábra).

2.4. UI

A UI (*User Interface*) foglalja magába az ágens szimulátor fő elemeit. A következőkben ennek az elemei, azok funkciói kerülnek felsorolásra.

A szimuláció figyelésére szolgáló UI ablakok, amelyek a szimuláció kezdete után a képernyőn jelennek meg, a következők:

- *Statistics*: A `C` billentyű megnyomására megjelenik az első számú ágens Statistics ablaka (2.4. ábra).

- *Inventory*: Az `I` billentyű megnyomására megjelenik az az adott ágens Inventory oldala (2.3. ábra).

Statistics

- Egy adott ágens adatának a kiolvasásának az eredményeit megjelenítő UI ablak.
- A szimulációban megállítástól függetlenül a `C` billentyű megnyomására jelenik meg és tűnik el.
- Megnyitásakor mindig a még szimulációban létező ágensek közül az első számú ágens adait fogja visszaadni.
- A vizsgálandó ágens elpusztulásakor a még létező ágensek közül az új első számú ágens adatait fogja visszaadni.
- Ezen az ablakon a következő ágens adatokat írja ki:
  - *Sorszám*: Az ágens sorszámát adja vissza.
  - *HP* (életerő): Az ágens aktuális életerejét adja vissza.
  - *Erő* (str) Az ágens aktuális str értékét adja vissza.
  - *Kitartás* (vit): Az ágens aktuális vit értékét adja vissza.
  - *Kitérés* (eva): Az ágens aktuális eva értékét adja vissza.
  - *Pontosság* (acc): Az ágens aktuális acc értékét adja vissza.

  Ezek a számok változhatnak az alapján, hogy milyen felszerelést hord az adott ágens.
- Új tárgy felvételénél, ha úgy ítéli, hogy jobb, mint az általa hordott azonos típusú tárgy, akkor lecseréli, és ennek megfelelően frissülnek a statisztikái.

Inventory

- Nem nyitható meg, ha nincs még megnyitva egy tetszőleges ágens *Statistics* ablaka (2.3. ábra).
- Megnyitásakor annak az ágensnek az *Inventory*-ja lesz látható, amelyiknek a *Statistics* oldala nyitva van.
- A *Statistics* oldalon az előre/hátra lépés esetén az *Inventory* az újonnan szemügyre vett ágens *Inventory*-ját fogja megjeleníteni a felhasználó számára.
- Az *Inventory*-ban a következő típusú tárgyak találhatóak meg:
  - *Armor* (páncél): Páncélként hordható tárgy, 4 alapstatisztikát tartalmaz.
  - *Helmet* (sisak): Sisakként hordható tárgy, 4 alapstatisztikát tartalmaz.
  - *Weapon* (fegyver): Fegyverként hordható tárgy, 4 alapstatisztikát tartalmaz.
  - *Key* (kulcs): Ajtó kinyitásához szükséges tárgy.
- Az *Inventory*-ban a `WASD` billentyűk lenyomásával választhatjuk ki, hogy mely *Inventory* helyet akarjuk megnézni.

- A fehér üres négyzet jelöli azt a helyett az *Inventory*-ban, amelyet éppen szemügyre vesz a felhasználó.
- Ha felszerelés típusú az adott tárgy, akkor a nevét és a 4 alapstatisztika értékét írja ki.
- Ha nem felszerelés típusú az adott tárgy, akkor pedig a nevét írja ki.

Életcsík

- Minden ágens fölött az aktuális életcsíkja jelenik meg automatikusan.

2.5. Szimulációs környezet specifikálása

A következőkben a szimulációs környezet modelljének, elemeinek a specifikálására kerül sor.

2.5.1. Map

A szimulációban az ágensek egy diszkrét négyzetrács felbontáson tudnak mozogni. Ennek a mérete rögzített, 50×50, és minden elemhez egy saját definiálású `block` típus tartozik.

Egy `block`-nak van típusa, amely egy egész számként reprezentálható. Ez a következő értékeket veheti fel:

- 0 típus: Üres, a *map*-on nem elérhető területeket jelöli.
- 1 típus: Fal, a bejárható területeket körbezáró fal, amelyen nem lehet átmenni.
- 2 típus: Füves terület, az ágensek által bejárható területek.

Az 50×50-es mátrixon minden 2-es típusú blokkra helyezhetünk el egy objektumot. Az objektumok típusai az alábbiak lehetnek.

- *Brick*: Egy mozgást korlátozó objektum. Nincs különösebb szerepe.
- *Chest*: *End screen*-t előidéző tárgy, amely a szimuláció végét jelenti.
- *Door*: Egy mozgást korlátozó objektum. Kulccsal kinyitható az ágensek által.

Az objektumok egy részhalmaza olyan, hogy az ágensek fel tudják azokat venni az Inventory-jukba, amelyek nem korlátozzák a mozgást. Ezek az alábbiak:

- *Key*,
- *Entity_ Armor_ 01*,
- *Entity_ Weapon_ 01*,
- *Entity_ Helmet_ 01*,
- *Entity_ Armor_ 02*,

- *Entity_ Weapon_ 02*,
- *Entity_ Helmet_ 02*.

Szimuláció kezdetekor megtörténik a *map* beolvasása. Ezt követően minden 2-es típusú blockra helyezhetünk el egy ágenst.

- *EntityFaction0*: 0-es számú csapatban lévő ágens.
- *EntityFaction1*: 1-es számú csapatban lévő ágens.

A szimulációs környezetben adott számú elemet helyezünk le. Konkrétan az alábbi számú és típusú elemek kerülnek rá:

- 18 szoba,
- 25 ajtó,
- 25 kulcs,
- 4 ágens,
- 10 felvehető tárgy,
- 2 chest,
- 12 brick.

2.5.2. Szobák és folyosók

A *map*-en szobák és folyosók alakíthatók ki, de nem tetszőleges módon. Az alábbi szabályrendszer írja le, hogy milyen feltételek szerint helyezheti le ezeket a program.

- A szobákat több szomszédos cellák alkotják.
- A szobákat folyosók kötik össze, amelyeket pontosan kettő cellával szomszédos cellák alkotnak.
- Minden szomszédos szobát egy ajtó választ el egymástól.
- A szobák celláin korlátozottan helyezkedhetnek el oszlopok, amely a cellát nem bejárható cellává alakítja át, és korlátozza a látást.
- Minden szoba tartalmaz legalább 1 kulcsot, amely elősegíti a *map* bejárhatóságát az ágensek által.

2.5.3. További map-re vonatkozó szabályok

A *map* szobák és folyosók összessége, ahol a szobák alakjánál törekszünk a négyzet alakú szobák elkerülésére. A folyosók általában rövidek és egy szobából akár több is nyílhat egy adott szobára. A *map* blokkokból épül fel. Blokkok típusát egy $[0, 2]$ intervallumon belüli számok jelölik. Ezek az adatok a szimuláció kezdetekor kerülnek beolvasásra a `world01.txt`-ből.

A szobákban és a folyosókban bejárható terület egy mátrixhoz hasonló, amely celláit blokkoknak nevezzük.

Minden blokkhoz tartoznak további adatok, úgy mint:

- *X* koordináta (Integer),
- *Y* koordináta (Integer).

Fontosabb kizáró feltételek:

- Ajtó csak szoba és a folyosó, vagy szoba és a szoba között létezzen.
- Folyosóból ne nyíljon ajtó folyosóra.
- Két különböző folyosó ne érjen össze.
- Nem lehet olyan cellára mozgást kizáró objektumokat letenni, amely lehetetlenné teszi a *map* egy tetszőleges pontjáról a *map* egy másik tetszőleges pontjára való eljutását.

2.6. Tárgyak

A tárgyakhoz a következő fontosabb adatok tartoznak:

- *name* (String),
- *typeName* (String),
- *type* (Integer),
- *str* (Integer),
- *vit* (Integer),
- *eva* (Integer),
- *acc* (Integer).

2.6.1. name

Az adott tárgyat bemutató név.

2.6.2. typeName

A tárgyak tartalmaznak tárgy típus nevet, például: Armor, Helmet, Key... stb.

2.6.3. type

A Key tárgy a "0" típusú, amíg minden hordható tárgy a "2" típus számot tartalmazza.

2.6.4. str

Ha az adott tárgy az ágensek által hordható tárgy, azaz nem kulcs, van valós str értéke.

2.6.5. vit

Ha az adott tárgy az ágensek által hordható tárgy, azaz nem kulcs, van valós vit értéke.

2.6.6. eva

Ha az adott tárgy az ágensek által hordható tárgy, azaz nem kulcs, van valós eva értéke.

2.6.7. acc

Ha az adott tárgy az ágensek által hordható tárgy, azaz nem kulcs, van valós acc értéke.

2.7. Tárgy nevek

2.7.1. Entity_Helmet_01

Ágens által használható felszerelési tárgy, amely statisztikát ad, amelyek hozzáadódnak, illetve kivonódnak az ágens jelenlegi statisztikáiból.

- Erő (str): [1,3[
- Kitartás (vit): [1,3[
- Kitérés (eva): [1,3[
- Pontosság (acc): [1,3[

2.7.2. Entity_Helmet_02

Ágens által használható felszerelési tárgy, amely statisztikát ad, amelyek hozzáadódnak, illetve kivonódnak az ágens jelenlegi statisztikáiból.

- Erő (str): [0,4[
- Kitartás (vit): [0,4[
- Kitérés (eva): [0,4[
- Pontosság (acc): [0,4[

2.7.3. Entity_Armor_01

Ágens által használható felszerelési tárgy, amely statisztikát ad, amelyek hozzáadódnak, illetve kivonódnak az ágens jelenlegi statisztikáiból.

- Erő (str): [1,3[
- Kitartás (vit): [1,3[
- Kitérés (eva): [1,3[
- Pontosság (acc): [1,3[

2.7.4. Entity Armor 02

Ágens által használható felszerelési tárgy, amely statisztikát ad, amelyek hozzáadódnak, illetve kivonódnak az ágens jelenlegi statisztikáiból.

- Erő (str): [0,4[
- Kitartás (vit): [0,4[
- Kitérés (eva): [0,4[
- Pontosság (acc): [0,4[

2.7.5. Entity Weapon 01

Ágens által használható felszerelési tárgy, amely statisztikát ad, amelyek hozzáadódnak, illetve kivonódnak az ágens jelenlegi statisztikáiból.

- Erő (str): [1,3[
- Kitartás (vit): [1,3[
- Kitérés (eva): [1,3[
- Pontosság (acc): [1,3[

2.7.6. Entity Weapon 02

Ágens által használható felszerelési tárgy, amely statisztikát ad, amelyek hozzáadódnak, illetve kivonódnak az ágens jelenlegi statisztikáiból.

- Erő (str): [0,4[
- Kitartás (vit): [0,4[
- Kitérés (eva): [0,4[
- Pontosság (acc): [0,4[

2.7.7. Kulcs

Celláról szerezhetőek meg. Szerepük az ajtók kinyitása az ágens által. Kulcsot tartalmazó ágens, elpusztulásakor a legutolsó helyén hagyja a kulcsot.

2.8. Ágensek

2.8.1. Statisztika

- Ágensek 4 alap statisztikával rendelkeznek.
- Ez a 4 statisztika a következők: Erő, Kitérés, Kitartás, Pontosság.
- Az ágensek kezdő statisztikája fix 1.

A következő tárgyakkal kezd a szimulációban:

- Entity Helm 01,
- Entity Armor 01,
- Entity Weapon 01.

2.8.2. Sebzés számlálás

A támadásnak két végkimenetele lehet:

- Találat,
- Eltévesztve.

Találat esetén, kiszámolódik a sebzés mértéke.
Eltévesztés esetén nem történik meg a sebzés.

2.8.3. Statisztikák jelentőssége

Statisztikák jelentőssége két különböző faction változóval rendelkező ágens összetalálkozásánál látszódik meg. Két fontosabb dologban játszik szerepet, az ágens statisztikája a fent említett találkozásnál.

Egy ágens, egy másik ágensre mért csapásának sikeressége függ attól, hogy nagyobb-e a támadó ágens acc értéke, mint a támadást elszenvedő fél eva értéke. Ha nagyobb a támadó ágens acc értéke, mint a támadott eva értéke, akkor 100%-osan betalál a sebzése, azaz támadás betalálása következik be. Ha nem nagyobb, akkor 20% esélye van arra a támadott félnek, hogy ne kapjon semmilyen sebzést, azaz támadás eltévesztése következik be.

Egy ágens, egy másik ágensre mért csapásának a sebzése függ attól, hogy nagyobb-e a támadó ágens str értéke, mint a támadást elszenvedő fél vit értéke. Ha nagyobb a támadó ágens str értéke, mint a támadott vit értéke, akkor 20% esélye van arra a támadó félnek, hogy dupla sebzést mérjen be, ekkor a támadás nagysága nagy. Ha nem nagyobb, akkor fixen 1 sebzést okoz a támadása, ekkor a támadás nagysága normál.

2.9. Ágens főbb tulajdonságai

- str (Integer)

  Az ágens létrejöttekor kerül megadásra általunk.

- vit (Integer)

  Az ágens létrejöttekor kerül megadásra általunk.

- eva (Integer)

  Az ágens létrejöttekor kerül megadásra általunk.

- acc (Integer)

  Az ágens létrejöttekor kerül megadásra általunk.

- maxLife (Integer)

  Az ágens létrejöttekor kerül megadásra általunk.

- faction: (Integer)

  Lehet 0,1. Fontos változó a combat létrejöttéhez.

- healTurn: (Integer)

  Változó, amely azt tárolja, hogy melyik kör végén lesz HP regeneráció.

2.9.1. Kinézetet befolyásoló tényező

- Ágens állása. Látszódik, hogy milyen irányba néz.
- Az ágens faction típusa.

2.9.2. Kommunikáció közöttük

Ágensek bemenete, információ szerzése:

- A hozzájuk szomszédos blockokat vizsgálják, ha nem találnak semmit megvizsgálják az általuk érzékelhető blockokat.
- A balra, jobbra, felfelé és lefelé irányban érzékelnek 2 blocknyi távolságra, ha nem gátolja meg valami az útjukat.

Ágensek belső memóriája a következőket tartalmazza:

- Inventory állapota.
- Karakter ablak állapota.
- Aktuális HP állapota.
- A blockok hashMap értékei. (hányszor volt az adott blockokon, segít felderíteni a teljes mapot.)
- Észlelt Objektumok helyzete, típusa.
- Inventoryban tárolt itemek.
- Észlelt ágens értékei. (ellenséges, vagy nem)
- Észlelt ágens helyzete. (X,Y koordináta)

2.9.3. Megfigyelési akcióik

Információk feldolgozása:

- Inventory mérete és kihasználtsága, ha elérte a max méretet, nem képes az általa kívánt Tárgyat felvenni.
- Karakter ablakban tárolt statisztikák számossága.

- Aktuális HP mennyiség. 5 Körönként 1 HP visszatöltés.(Lásd: Idő)
- Az adott körben lehetséges lépesek hashMap értékei vizsgálata, amelyek az adott block koordinátáit tárolják és azt, hogy az adott ágens hányszor lépett rá az adott blockra. (Szükséges azért, hogy idővel biztosan felfedezze az egész mapot)
- Észlelt objektumok helyzete, típusa. Ezekből az információkból eldönti a prioritást és maghatározza, hogy mely irányba fog végül haladni.
- A nem használt tárgyak törlése, ha nem jobb a jelenleg hordottnál, ezáltál biztosítva, hogy ne teljen meg az inventory.
- Ágens értékeiből leszűrt információk számításba vétele. (ellenség-e)
- Támadható ágensek közül a legkisebb HP-val rendelkező ágens támadását kezdeményezi.

2.9.4. Ágens tevékenységeinek a köre

Ezek bármelyik használata, felhasználja az ágens körét.

- Lépés.
  - A lépés akkor lehetséges, hogyha az ágens közvetlen közelében, azaz vagy az X vagy az Y koordinátájával szomszédos cellán nem tartózkodik mozgást korlátozó ágens/térbeli objektum. Lásd itt: Mapon található mozgást kizáró objektumok
- Ajtó nyitás.
  - Szükséges az Inventory-ban egy Key nevű tárgy, amely a kulcs.
  - Ha az ágens közvetlen közelében létezik ajtó, a nyitás akkor lehetséges.
  - Nyitása után eltűnik az ajtó, és a kulcs az Inventory-ból.
- Chest nyitás.
  - Ha az ágens celláján létezik az ellenkező faction chest-je, akkor a nyitás lehetséges.
  - Ha nincs támadható ágens, akkor kinyitja.
  - Megjelenő képernyő kinyitása esetén, ha piros: 2.7, ha kék: 2.6
- Felvétel.
  - Ha az Inventory-ban van hely, akkor lehetséges.
  - Ha az ágens celláján létezik valamilyen tárgy, ekkor a tárgyat a felvétel akcióval eltünti a celláról és az Inventory-ba kerül.
- Törlés.

  - Ha az Inventory-ban van olyan tárgy, amely nem kulcs és nem hordott az ágens által, akkor lehetséges.
  - Az ágens Inventory-jából kitörli az tárgyat.

- Felszerelés.
  - A felszerelés mindig lehetséges az ágens saját körében.
  - Megvizsgálja, hogy az utoljára felvett tárgy statisztikái jobbak-e, mint az ő általa hordott azonos típusú hordható tárgy.
  - Ha igen, felveszi azt, ezáltal megváltoztatva saját statisztikáit.

- Támadás.
  - A támadás akkor lehetséges, hogyha az ágens közvetlen közelében, azaz vagy az X vagy az Y koordinátájával szomszédos cellán tartózkodik egy ellenséges ágens.
  - Több ellenséges ágens esetén, a legkisebb HP-val rendelkező fogja támadni.

2.9.5. Ágens fő céljai

- Megtalálni az ellenséges chestet.
- Elpusztítani az ellenséges ágenseket.

2.9.6. Ágens másodlagos céljai

- Tárgy felvétel,
- Tárgy hordása adott esetekben,
- Ajtó kinyitás,
- Inventory menedzsment,
- Mindig a kevesebbszer bejárt blockokat választani mozgásakor.

2.9.7. Ágens függvény

A következő ábrán látható az ágens cselekvésének meghatározására szolgáló ciklus, amely a szimuláció kezdetétől működik. 2.8

A csomópontok fontossági sorrendben vannak felsorolva.
A csomópontok kifejtése:

- wantToAttack?
  - Ha közvetlen közelében van egy ellenséges ágens, mindig azt az akciót választja, ahol támadást mér be az ellenséges ágensnek.

- wantToEquip?

  - Ha a legutoljára általa felvett tárgy statisztikái több statisztikát adnak összességében, mint az általa hordott azonos típusú tárgy, akkor lecseréli.

- wantToPickUp?

  - Ha az ágenset tartalmazó block-on létezik valamilyen felvehető tárgy az ágens által, akkor felveszi.

- wantToDeleteAnItem?

  - Ha az ágens Inventory-ja tartalmaz olyan hordható tárgyat, amely nem jobb mint az általa hordott tárgy, akkor eltávolítja az inventoryjából.

- wantToOpenDoor?

  - Ha van kulcs nála és a közvetlen közelében van egy ajtó, akkor az ajtó kinyitását választja.

- wantToMoveToItem?

  - Ha van olyan közvetlen közeli block az ágens szomszédjában, amelyre lehetséges a lépés és tartalmaz valamilyen felvehető tárgyat, akkor összeveti az ágens és a felvenni kívánt tárgy koordinátáit, és az alapján eldönti milyen irányba mozogjon az ágens.

- newEnemy?

  - Ez az eset akkor lép fel, ha semmilyen más cselekvést nem akart elvégezni az adott körben az ágens és lát a detectableBlocks-on belül olyan blockot, amely egy ellenséges ágenst tartalmaz, ilyenkor a hozzá közeli blockot választja ki következő lépésének.

- newItem?

  - Ez az eset akkor lép fel, ha semmilyen más cselekvést nem akart elvégezni az adott körben az ágens és lát a detectableBlocks-on belül olyan blockot, amely egy kívánt tárgyat tartalmaz, ilyenkor a hozzá közeli blockot választja ki következő lépésének.

- wantToMove?

  Ez az eset akkor lép fel, ha semmilyen más cselekvést nem akart elvégezni az adott körben az Ágens.

  - A lehetséges blockok közül kiválasztja azt a blockot a következő lépésének, amelyen még nem volt soha.
  - Ha több olyan lehetséges block van, amelyre léphet, akkor véletlenszerűen választ egyet a lehetőségek közül.
  - Ha nincs egy olyan block se, ahol ne lett volna már, akkor a legkevesebbszer bejárt blockot fogja választani.
  - Ha több legkevesebbszer bejárt block létezik a lehetséges blockok közül, akkor véletlenszerűen választ egyet a lehetőségek közül.

2.10. Cél Tárgy

Map teljesítéséhez szükséges tárgy, amely egy chest. A kék faction-nek a piros chestet, a piros faction-nek a kék chestet kell elérnie. A mapon egy szobában helyezkedik el.

2.11. Combat

A szimulációban 1-es faction változójú ágens és 2-es faction változójú ágens közötti HP elvétel lehetséges,

- szomszédos cellából támadás által,
- a támadás lehet betalált, elvétett,
- a támadás nagysága lehet normális vagy nagy.

2.12. Interface felület

2.12.1. Pause

- prekondíció: A szimulációs ablakban vagyunk.
- általános működés: Megnyomjuk a P gombot.
- alternatív esetek: Rossz tevékenység történik be.
- Postkondíció: Megjelent a Paused kiírás bal fent.
- kivételes esetek: Nem ált meg a játékmenet.

2.12.2. Inventory

- prekondíció: A szimulációs ablakban vagyunk és meg van már nyitva a karakter ablak.
- általános működés: Megnyomjuk az I gombot.
- alternatív esetek: Rossz ablak nyílik meg.
- Postkondíció: Megnyílt az inventory ablak.
- kivételes esetek: Nem nyílt meg az ablak.

2.12.3. Karakter

- prekondíció: A szimulációs ablakban vagyunk.
- általános működés: Megnyomjuk a C gombot.
- alternatív esetek: Rossz ablak nyílik meg.
- Postkondíció: Megnyílt a Karakter ablak.
- kivételes esetek: Nem nyílt meg az ablak.

2.13. Mapon található mozgást kizáró objektumok

2.13.1. Brick

A Brick egy oszlop, amely a map beolvasása után kerülnek bizonyos cellákra. Oszlopoknak semmilyen különleges funkciója nincs azonkívül, hogy map komplexitását kívánja növelni azzal, hogy mozgást és látást korlátozó szerepet lát el.

2.13.2. Zárt ajtó

Az ajtók a map beolvasása után kerülnek bizonyos cellákra. Az ajtók mozgást korlátozó szerepet töltenek be, ha nincs az ágensnél kulcs. Ha van, akkor szomszédos blockból kinyithatja az ajtókat. Ahogyan az brickek, a zárt ajtók is map komplexitását kívánja növelni.

2.14. Idő

Az idő a szimuláción belül minden ágens által végrehajtott akció által telik. Minden 5. kör után, 1 HP-t töltenek vissza az ágensek, ha kisebb az aktuális életerejük, mint a maximum. Ha egy ágens elvégez egy akciót, akkor az irányítás átkerül egy másik ágensre.

2.15. Kamera

A kamera block-ról blockra képes haladni 3 koordináta meglépésével. A kamera képes mozogni, akkor is ha a jáétkmenet szünetel.(Paused)

2.16. Irányítás

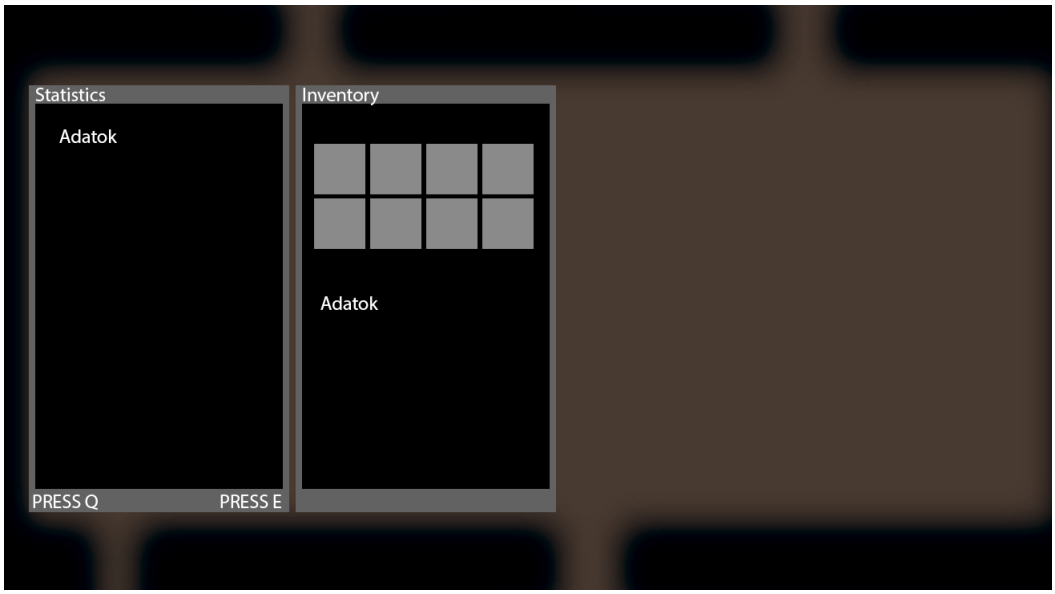
Szimulációban használható irányítás:

- Kamera mozgatása: WASD
- Ágens Statistics ablak megnyitása/bezárása: C
- Ágensek Statistics ablaka közötti lépegetés: Q(hátra), E(előre)
- Adott ágens Statistics oldalhoz Inventory megnyitása/bezárása: I

2.17. Látványterv képek, használt képek és diagramok

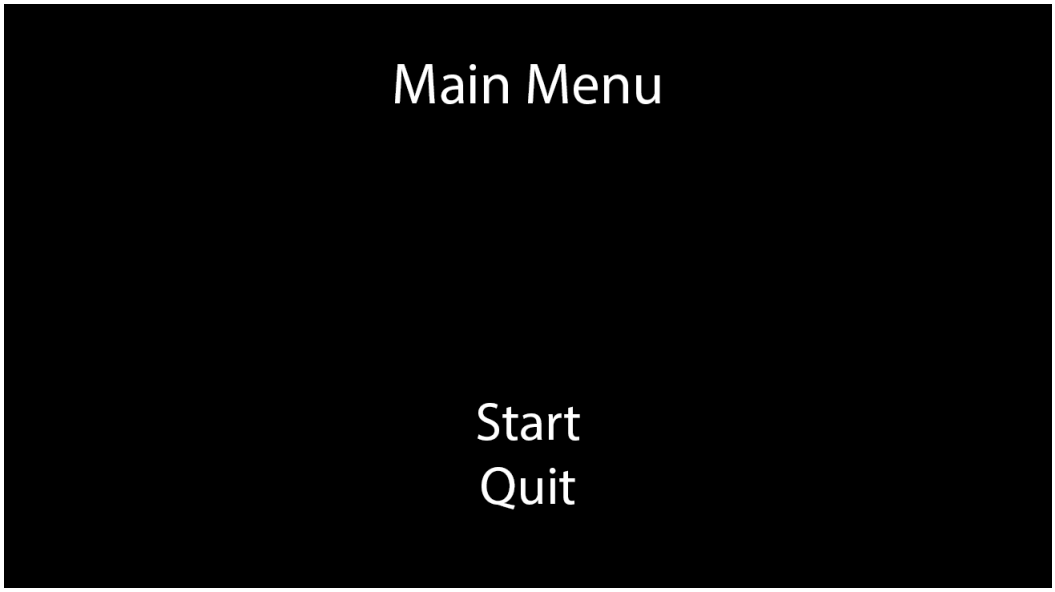
2.1. ábra. nonPaused

2.2. ábra. Paused

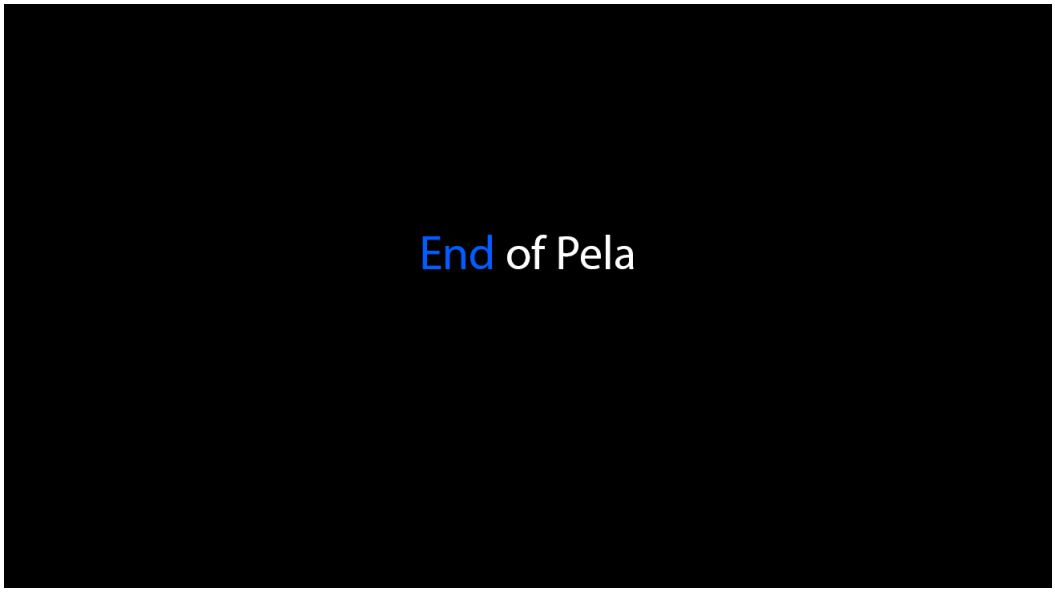


2.3. ábra. Inventory

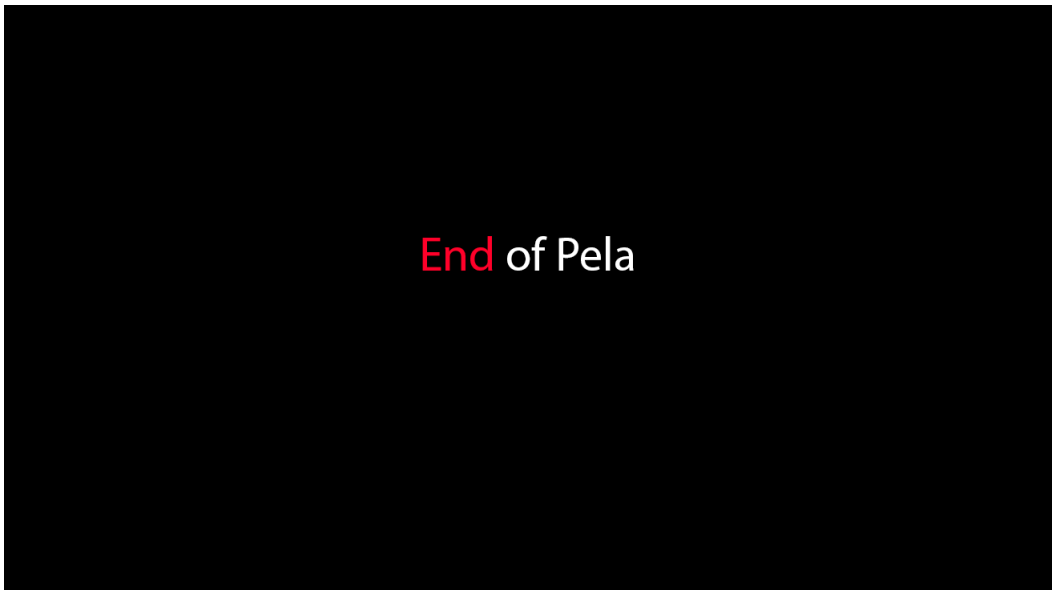
2.4. ábra. Statistics



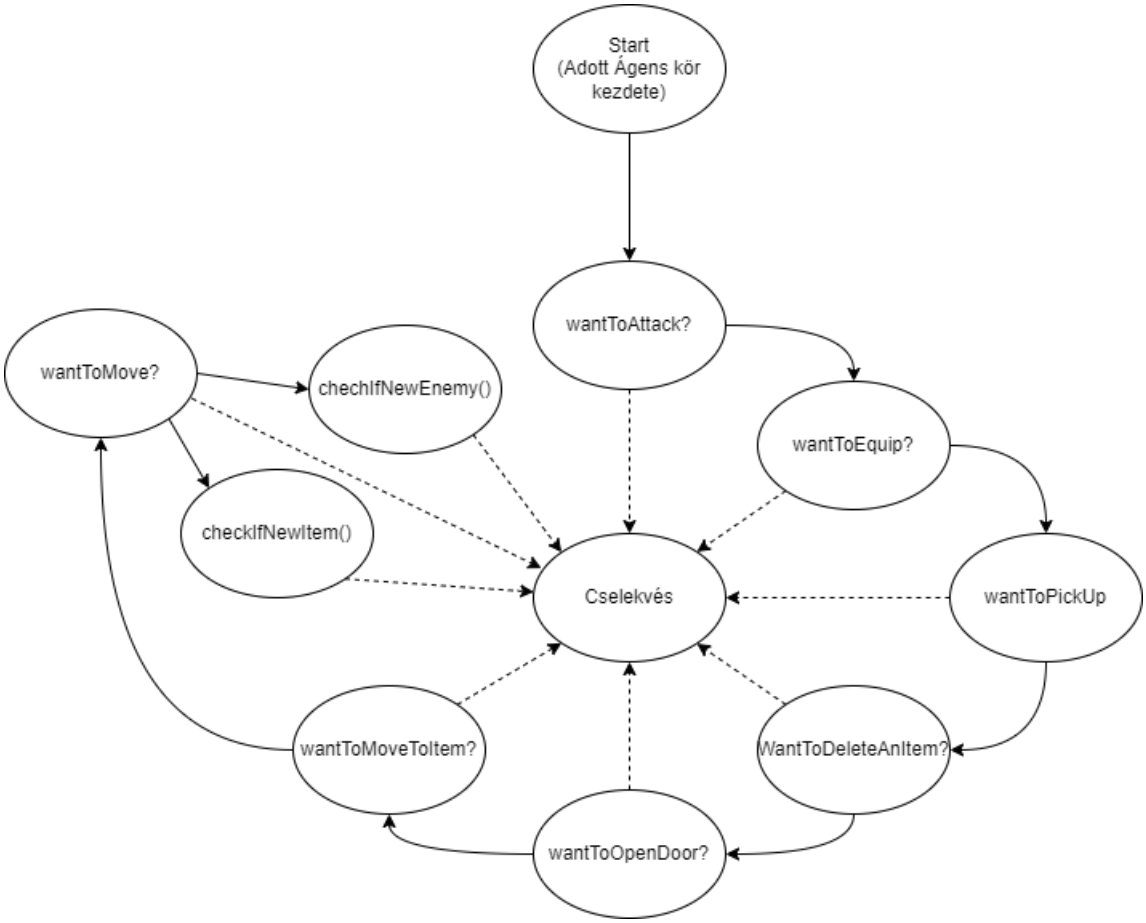
2.5. ábra. Main Menu



2.6. ábra. Blue End Screen

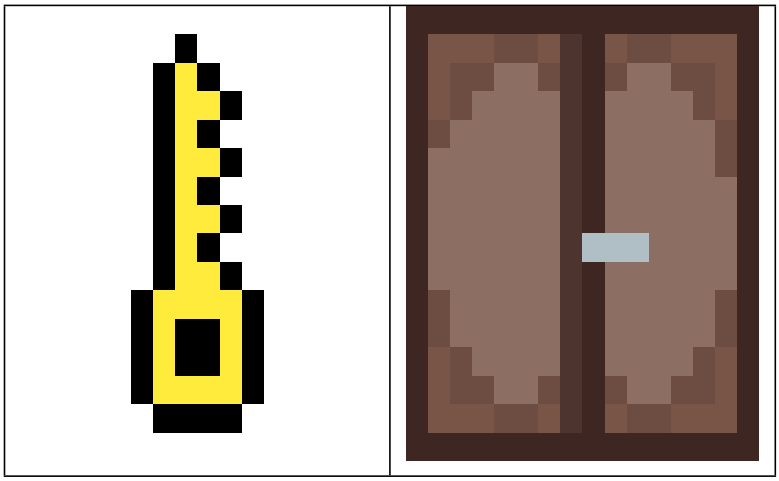


2.7. ábra. Red End Screen

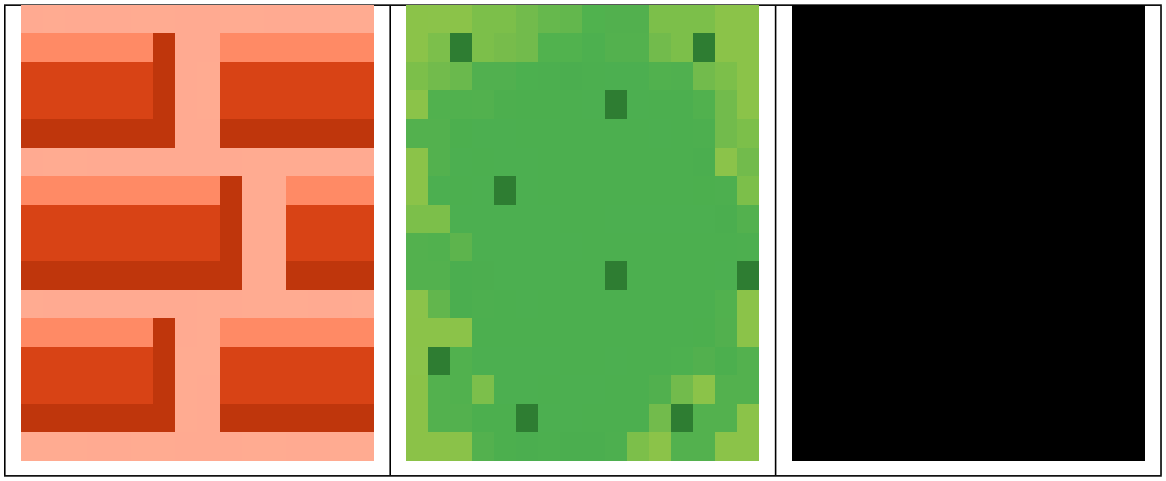


2.8. ábra. Agent Action Priority

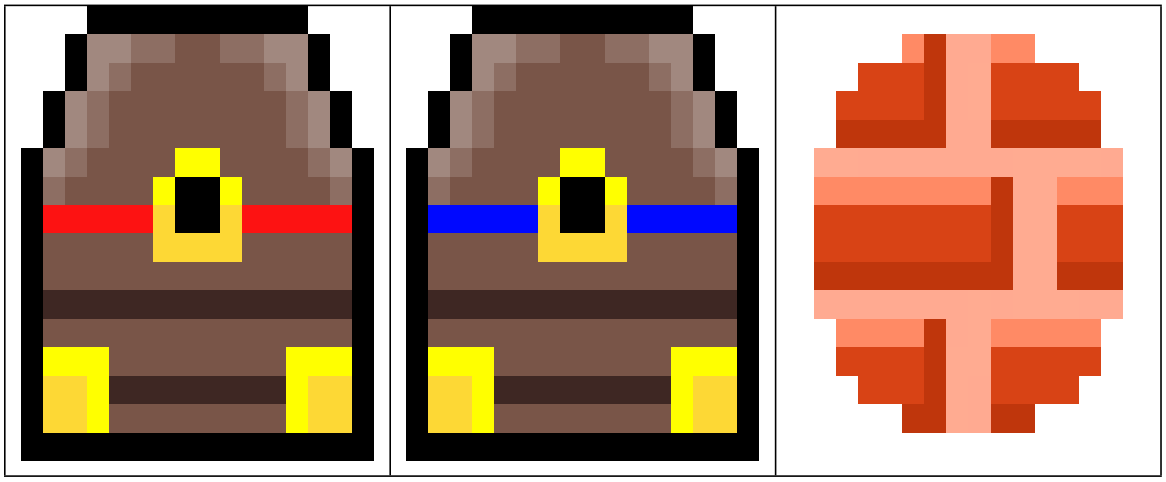
2.1. táblázat. Item set 01



2.2. táblázat. Key and Door



2.3. táblázat. Blockok



2.4. táblázat. Chests and Brick

3. fejezet

Tervezés

3.1. UML

3.1.1. KeyHandler

A KeyHandler osztály felel a kamera mozgatásáért, a játékmenet megállításért/elinditásáért, UI elemek használatáért, mint például az Inventory és Statistics oldal megnyitása/bezárása.

3.1.2. Block

Blockok néhány változóit tartalmazza.

3.1.3. BlockManager

Blockok típus definiálását, és a map beolvasást tartalmazza.

3.1.4. GamePanel

A játékmenet futását biztosítja.

3.1.5. Camera

Kamera kezdő helyzetét, lépés távját határozza meg.

3.1.6. AssetSetter

Az objektumok és ágensek elhelyezésért felelős.

3.1.7. UI

A gameScreenNumber változó által tárolt 3 fő képernyő megjelenítéséért felelős: Kezdő képernyő(title), játékmenet(normal) és végsőképernyő(end). Itt történnek a képernyőre való szöveg kiírások is. Inventory és Statistics ablak megjelenítését is tartalmazza.

3.1.8. Entity

Röviden az ágensek útvonalválasztásáért felelős. Itt történik az életcsík megrajzolása is. Tárolja az ágensek változóit.

3.1.9. ItemDecider

Entity osztálynak ad vissza igaz/hamis értékeket, amelyek befolyásolják az útvonalválasztást.

3.1.10. Object

Az objectek osztálya, object típusokat tartalmaz.

3.1.11. Entity1

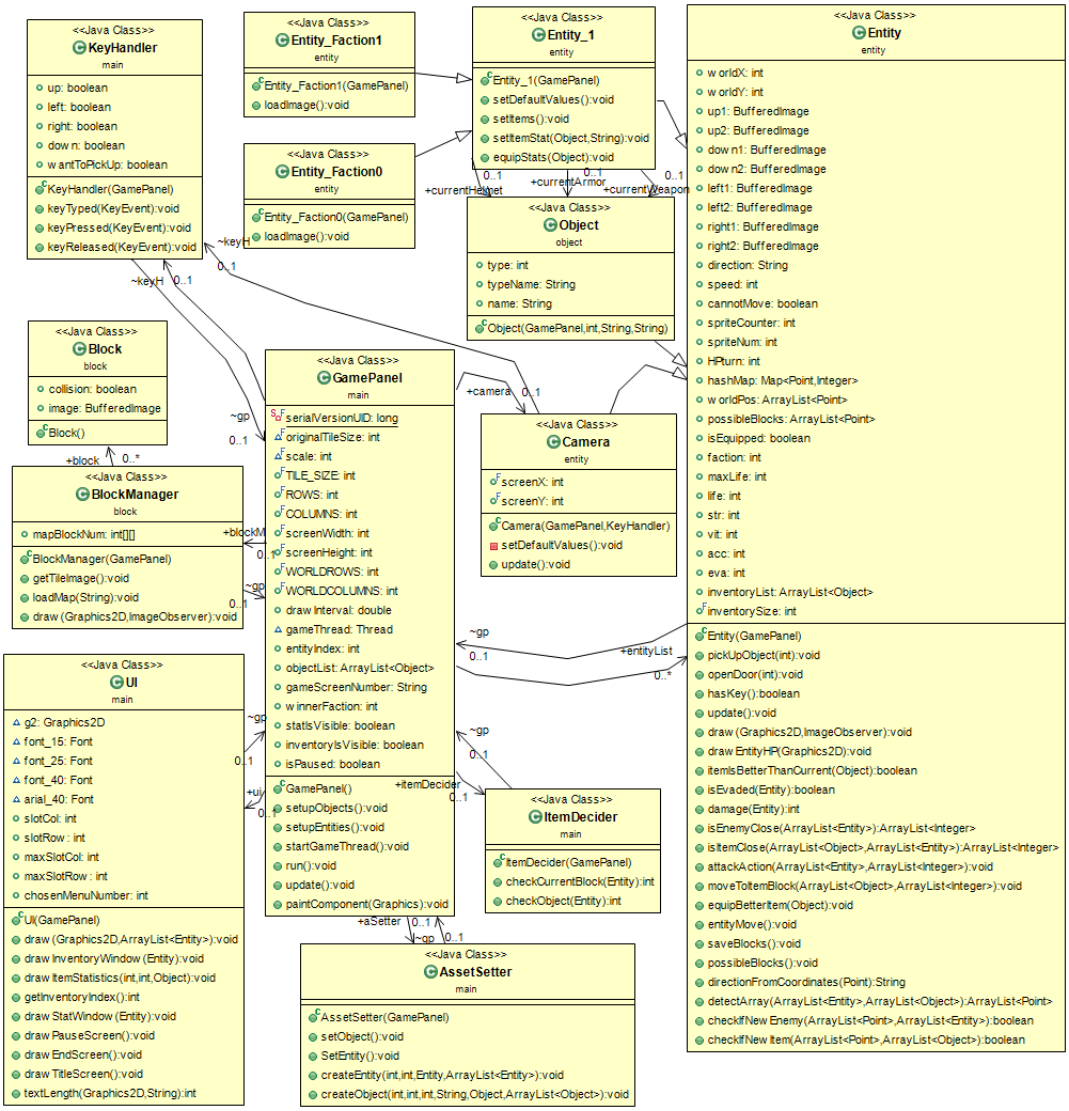
Ágensek egyetlen típúsának alap értékeinek definiálására szolgál.

3.1.12. EntityFaction1

A faction 1-hez tartozó ágensek képeit tárolja, amelyeket láthatunk a szimulációban.

3.1.13. EntityFaction0

A faction 0-hez tartozó ágensek képeit tárolja, amelyeket láthatunk a szimulációban.


<<Java Class>>
KeyHandler
main
up: boolean
left: boolean
right: boolean
down: boolean
wantToPickUp: boolean
KeyHandler(GamePanel)
keyTyped(KeyEvent):void
keyPressed(KeyEvent):void
keyReleased(KeyEvent):void
<<Java Class>>
Entity_Faction1
entity
Entity_Faction1(GamePanel)
loadImage():void
<<Java Class>>
Entity_Faction0
entity
Entity_Faction0(GamePanel)
loadImage():void
<<Java Class>>
Entity_1
entity
Entity_1(GamePanel)
setDefaultValues():void
setItems():void
setItemStat(Object,String):void
equipStats(Object):void
<<Java Class>>
Entity
entity
worldX: int
worldY: int
up1: BufferedImage
up2: BufferedImage
down1: BufferedImage
down2: BufferedImage
left1: BufferedImage
left2: BufferedImage
right1: BufferedImage
right2: BufferedImage
direction: String
speed: int
cannotMove: boolean
spriteCounter: int
spriteNum: int
HPturn: int
hashMap: Map<Point,Integer>
worldPos: ArrayList<Point>
possibleBlocks: ArrayList<Point>
isEquipped: boolean
faction: int
maxLife: int
life: int
str: int
vit: int
acc: int
eva: int
inventoryList: ArrayList<Object>
inventorySize: int
Entity(GamePanel)
pickUpObject(int):void
openDoor(int):void
hasKey():boolean
update():void
draw (Graphics2D,ImageObserver):void
drawEntityHP(Graphics2D):void
itemIsBetterThanCurrent(Object):boolean
isEvaded(Entity):boolean
damage(Entity):int
isEnemyClose(ArrayList<Entity>):ArrayList<Integer>
isItemClose(ArrayList<Object>,ArrayList<Entity>):ArrayList<Integer>
attackAction(ArrayList<Entity>,ArrayList<Integer>):void
moveToItemBlock(ArrayList<Object>,ArrayList<Integer>):void
equipBetterItem(Object):void
entityMove():void
saveBlocks():void
possibleBlocks():void
directionFromCoordinates(Point):String
detectArray(ArrayList<Entity>,ArrayList<Object>):ArrayList<Point>
checkIfNewEnemy(ArrayList<Point>,ArrayList<Entity>):boolean
checkIfNewItem(ArrayList<Point>,ArrayList<Object>):boolean
<<Java Class>>
Object
object
type: int
typeName: String
name: String
Object(GamePanel,int,String,String)
+currentHelmet
+currentArmor
+currentWeapon
~keyH
<<Java Class>>
Block
block
collision: boolean
image: BufferedImage
Block()
<<Java Class>>
GamePanel
main
serialVersionUID: long
originalTileSize: int
scale: int
TILE_SIZE: int
ROWS: int
COLUMNS: int
screenWidth: int
screenHeight: int
WORLDROWS: int
WORLDCOLUMNS: int
drawInterval: double
gameThread: Thread
entityIndex: int
objectList: ArrayList<Object>
gameScreenNumber: String
winnerFaction: int
statIsVisible: boolean
inventoryIsVisible: boolean
isPaused: boolean
GamePanel()
setupObjects():void
setupEntities():void
startGameThread():void
run():void
update():void
paintComponent(Graphics):void
+camera
<<Java Class>>
Camera
entity
screenX: int
screenY: int
Camera(GamePanel,KeyHandler)
setDefaultValues():void
update():void
+block
<<Java Class>>
BlockManager
block
mapBlockNum: int[][]
BlockManager(GamePanel)
getTileImage():void
loadMap(String):void
draw (Graphics2D,ImageObserver):void
+blockM
~gp
+entityList
<<Java Class>>
UI
main
g2: Graphics2D
font_15: Font
font_25: Font
font_40: Font
arial_40: Font
slotCol: int
slotRow: int
maxSlotCol: int
maxSlotRow: int
chosenMenuNumber: int
UI(GamePanel)
draw (Graphics2D,ArrayList<Entity>):void
drawInventoryWindow (Entity):void
drawItemStatistics(int,int,Object):void
getInventoryIndex():int
drawStatWindow (Entity):void
drawPauseScreen():void
drawEndScreen():void
drawTitleScreen():void
textLength(Graphics2D,String):int
+itemDecider
<<Java Class>>
ItemDecider
main
ItemDecider(GamePanel)
checkCurrentBlock(Entity):int
checkObject(Entity):int
+aSetter
<<Java Class>>
AssetSetter
main
AssetSetter(GamePanel)
setObject():void
SetEntity():void
createEntity(int,int,Entity,ArrayList<Entity>):void
createObject(int,int,int,String,Object,ArrayList<Object>):void

3.1. ábra. UML

4. fejezet

Megvalósítás

Ebben a fejezetben a ágens heurisztikus mozgását és akcióit végrehajtó biztosító függvények bemutatása történik meg röviden.

```
    if(wantToAttack) {
        attackAction(gp.entityList,isEnemyCloseIndexList);
    }else if(wantToEquip) {
        equipBetterItem(inventoryList.get(inventoryList.size()
        -1));
    }else if(wantToPickUp) {
        pickUpObject(curObjIndex);
    }else if(wantToDeleteAnItem) {
        System.out.println("Deleted item: "
            + inventoryList.get(deleteIndex).name);
        inventoryList.remove(deleteIndex);
    }else if(wantToOpenDoor) {
        openDoor(doorIndex);
    }else if(wantToMoveToItem) {
        if(cannotMove==false) {
            moveToItemBlock(gp.objectList,
            isItemCloseIndexList);
            entityMove();
        } else System.out.println("Skipped Turn");

    }else if(wantToMove) {
        if(cannotMove==false) {

            ArrayList<Point> detectableBlocks =
                 detectArray(gp.entityList,gp.objectList);

            if(!checkIfNewEnemy(detectableBlocks,
            gp.entityList)) {
                if(!checkIfNewItem(detectableBlocks,
                gp.objectList)) {
                    possibleBlocks();
                }
```

```
            }

            entityMove();
        } else System.out.println("Skipped Turn");


    }
```

4.1. wantToAttack

Boolean típusú változó, amely akkor igaz, ha közvetlen szomszédos blockokon létező ellenséges ágensek száma nagyobb, mint 0.

Igaz érték esetén meghívódik az attackAction() függvény, amelynek paramétere az ágensekből álló lista és az isEnemyCloseIndexList, amely a közeli blockon lévő ellenséges ágensek közül a legkevesebb HP-val rendelkező ágens indexét adja vissza az ágens listából.

Az attackAction() függvény beállítja az adott ágens irányát, arra amelyik irányba hajta végre a támadását és elvégzi a támadást.

Támadás előtt először megvizsgálja betalált-e a találat, ha betalált, akkor meghívódik a damaga() függvény, amely kivonja az adott sebzést az adott ágenstől, amelyik elszenvedte a támadást.

4.2. wantToEquip

Boolean típusú változó, amely akkor igaz, ha az adott ágens inventoryjában az utoljára felvett tárgy hordható típusú és összességében több statisztikát ad, mint az ágens által jelenleg hordott azonos típusú tárgy.

Az equipBetterItem() függvénynek egy paramétere van, amely egy Integer szám, ami az adott ágens inventoryjának utolsó használt slotja, amely a legutoljára felvett tárgy. Meghívásakor leszereli az eddig hordott tárgyat, és felszereli az újonnan felvett tárgyat. Statisztikák ablakon nyomon lehet követni ezt a változást.

4.3. wantToPickUp

Boolean típusú változó, amely akkor igaz, ha létezik valamilyen tárgy az ágens jelenlegi blockján.

Ha igaz értéket ad vissza, akkor meghívódik a pickUpObject() függvény, amelynek paramétere az adott blockon lévő tárgy indexe a tárgyak listájában. Meghívásakor az adott tárgy bekerül az ágens első nem használt inventory slotjába.

4.4. wantToDeleteAnItem

Boolean típusú változó, amely akkor igaz, ha létezik valamilyen nem viselt tárgy az ágens inventoryjában.

Igaz érték esetén megtörténik a deletedIndex által tárolt elsőnek talált nem használt tárgy indexének a törlése az inventoryjából, amely nem az ágens által hordott és nem kulcs.

4.5. wantToOpenDoor

Boolean típusú változó, amely akkor igaz, ha létezik ajtó az ágens szomszédos blockjában és van az ágens inventoryjában legalább 1 kulcs.

Az openDoor() függvénynek egy paramétere van, amely az ajtó indexe a tárgy listában. Meghívásakor az ajtó objektum eltűnik, és eltávolítja a felhasznált kulcsot az ágens inventoryjából.

4.6. wantToMoveToItem

Boolean típusú változó, amely akkor igaz, ha a közeli blockokon lévő tárgyak listája nagyobb, mint 0 és az ágens inventoryja nincs tele.

Valódi ágens mozgás csak akkor jön létre, ha van olyan szomszédos block, amelyre képes lépni, ezt a CannotMove false értéke biztosítja.

Majd meghívódik a moveToItemBlock(), amelynek két paramétere van, az objektum lista és a szomszédos közelében lévő tárgyak indexének a listája. Ez a függvény beállítja az ágens direction értéket.

Majd meghívódik az entityMove() függvény, amely az ágens által választott irányba lép előre egyet és vagy felrakja a hashMap-re, vagy növeli az értékét.

4.7. wantToMove

Boolean típusú változó, amely mindig igaz, ha minden fentebb sorolt boolean változú hamis, ebben a pontban összefoglalt akció fog megtörténni.

Valódi ágens mozgás csak akkor jön létre, ha van olyan szomszédos block, amelyre képes lépni, ezt a CannotMove false értéke biztosítja.

A pontokat tároló listában a detectArray() függvény minden olyan x,y párost átad, amely az ágenstől vízszintes vagy függőlegesen 2 blocknyira van és tartalmaz valamilyen kívánt tárgyat. A detectArray két paramétere az ágensek listája és az objektum lista.

A checkIfNewEnemy() függvény paramétere a pontokat tároló lista és az ágensek listája. Ha talál valamilyen ellenséges ágenst a vizsgálandó pontokon, akkor azt az irányt fogja beállítani az ágensnek, amely az ellenséges ágens felé néz.

Utána meghívódik az entityMove() függvény, amely a kiválasztott irányba lép egyet és vagy felrakja a hashMap-re, vagy növeli az értékét.

Ha a checkIfNewEnemy() hamis értéket ad vissza, akkor megvizsgálja a checkIfNewItem() függvényt, amelynek két paramétere van, a vizsgálandó pontokat tároló lista és az objektumok listája.

Ha talál valamilyen számára érdekes tárgyat a vizsgálandó pontokon, akkor azt az irányt fogja beállítani az ágensnek, amely a kívánt tárgy felé néz.

Ha a checkIfNewItem() függvény is hamis értéket ad vissza, akkor meghívódik a possibleBlocks() függvény, amely felel a map felfedezéséért.

A függvény megvizsgálja a lehetséges lépéseket, hogy van-e köztük olyan block, amelyen még egyszer sem járt az ágens. Ha több ilyen van, akkor véletlenszerűen választ egyet azok közül.

Ha már minden lehetséges blockon járt legalább egyszer, akkor a lehetséges lépések közül a legkevesebbszer bejárt blockot választja. Ha több ilyen block is van, amin ugyanannyiszor volt, akkor véletlenszerűen választ közülük egyet.

5. fejezet

Tesztelés

5.1. Az alkalmazás elindítása

Az alkalmazás elindításakor megjelenik a Main Menu kiírás, ezen az oldalon, közép lent lévő ként opció közül választhatunk, a fel és lefele nyilak segítségvel, a kiválasztott menü pont alatt egy aláhúzás fog megjelenni, amely a felhasználó számára jelzi, hogy melyik menü pont a jelenleg kiválasztott.

A számunkra megfelelő menüpont kiválasztása utána az enter lenyomásával véglegesíthetjük döntésünket. Ekkor a menüpontnak megfelelő utasítás fog végrehajtódni.

Ha a Start menüpontot választottuk, akkor elhagyjuk a main menüt és elkezdődik a szimuláció. Ha a Quit menüpontot választottuk, akkor az alkalmazás bezáródik.

5.2. Szimuláció közben használható funkciók

5.2.1. Pause

A szimuláció közben, a felhasználónak lehetősége van megállítani a szimulációt, hogyha megszeretne valami vizsgálni vagy csak megszeretné állítani ideiglenesen a szimulációt. Ezt a P gomb megnyomásával teheti meg.

Az alkalmazás a szimuláció megállását, a bal felső sarokban kiírt fehér Paused felirattal jelzi a felhasználó számára. Ha a szimuláció jelenleg meg lett állítva, akkor a P gomb még egyszer megnyomásával újra elindíthatjuk azt.

5.2.2. Ágens ablak

A szimuláció szüneteltésétől függetlenül bármikor megnyithatjuk az ágensek oldalát (Statistics), ahol mindig az első ágens adait fogjuk látni először. Ez az ablak 6 darab sort tartalmaz, mindegyik az ágens egy adatát írja ki a felhasználó számára. Az ágensek között a Q és E gombok megnyomásával haladhatunk hátra és előre. Az alkalmazás ezek használatára az ágens ablak bal alsó és jobb alsó sarkában lévő PRESS Q és PRESS E kiírásokkal vonja fel a felhasználó figyelmét.

5.2.3. Inventory ablak

Az ágens ablakban megjelenített ágensnek sorszámától és a szimuláció szüneteltésétől függetlenül bármikor megnyithatjuk a hozzá tartozó inventory ablakot (Inventory). Ez

az ablak tartalmaz 8 négyzetet és egy rövid tárgy leírást, amely lehet 1 vagy 6 sorú, a tárgy típúsától függően. Ha a tárgy hordható típusú, akkor első sorában a nevét, a további 4 sorában a statisztikáit láthatjuk, majd végül az utolsó sorában azt hogy fel van-e szerelve az ágens által. Ha a tárgy nem hordható (kulcs), akkor a tárgy leírásában csak az első sorát, a nevét fogjuk látni. Az ablak megnyitásakor mindig a bal felső négyzet van kijelölve, amelyet az alkalmazás úgy jelöl a felhasználó számára, hogy egy fehér négyzetet helyez rá. A négyzetekben tárgyak képeit láthatjuk, hogy ha a vizsgált ágens inventory listája tartalmaz valamilyen tárgyat. Ezt az ablakot az ágens ablakkal ellentétben nem lehet léptetni, mindig az ágens ablakban vizsgált ágens inventory adatait fogja kiírni. Ezt az ablakot önmagában nem lehet megnyitni, csak akkor ha már az ágens ablak meg van nyitva.

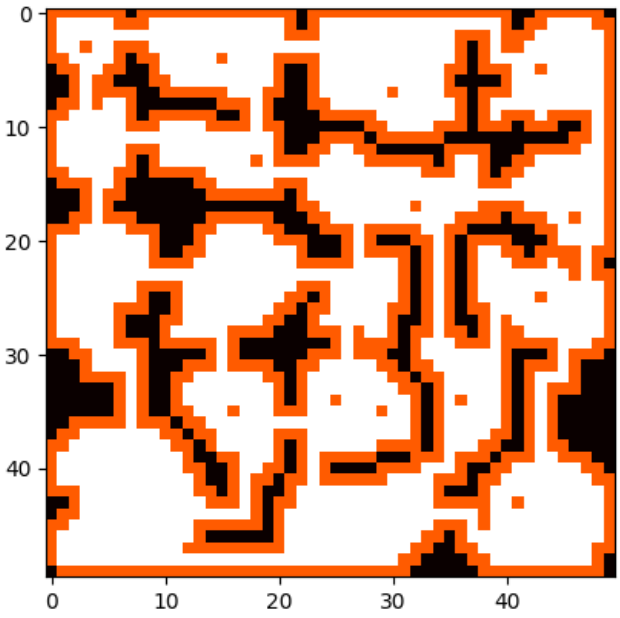
5.3. Szimuláció vége

A szimuláció befejződésekor a képernyő közepén láthatunk egy kiírást, amelyben a szín attól függ, hogy mely csapat nyerte a szimulációt.

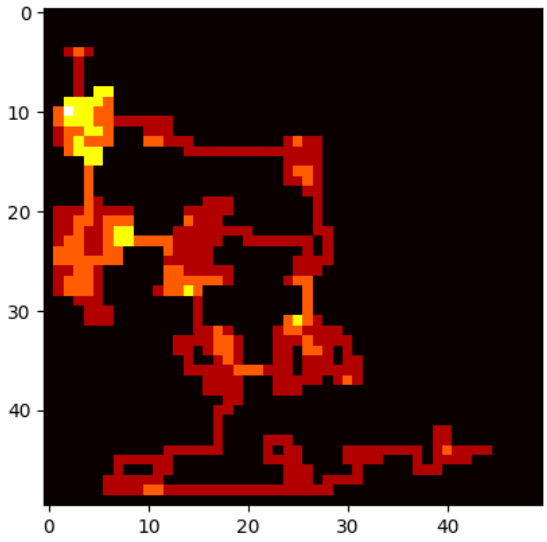
5.4. Eredmények

Itt látható, hogy mely ágens mely szimulációban milyen utat járt be a mapon, illetve a mapon lévő bejárható blockok is láthatóak a mapon.

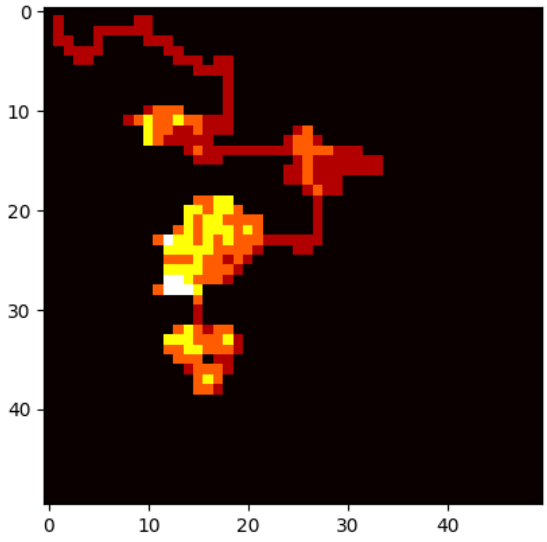
Két szimulációban kapott eredményeket mutatom be a következő képeken.



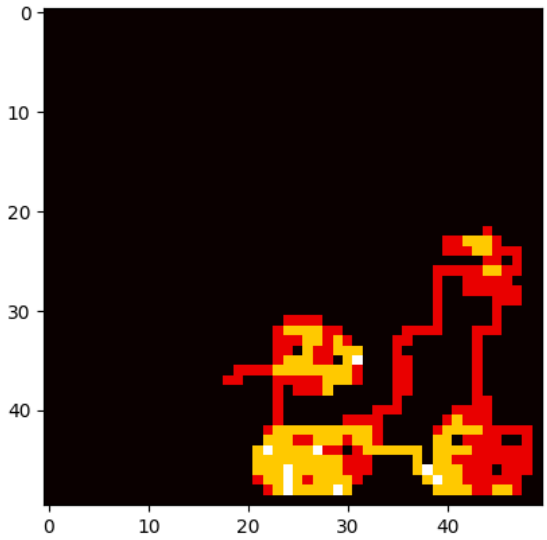
5.1. ábra. Map



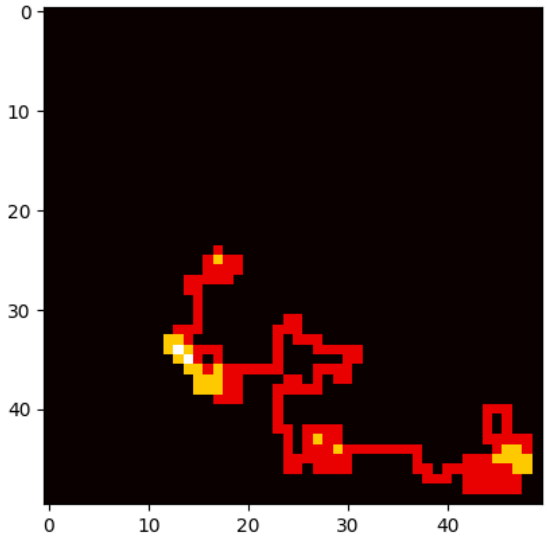
5.2. ábra. Agent 1



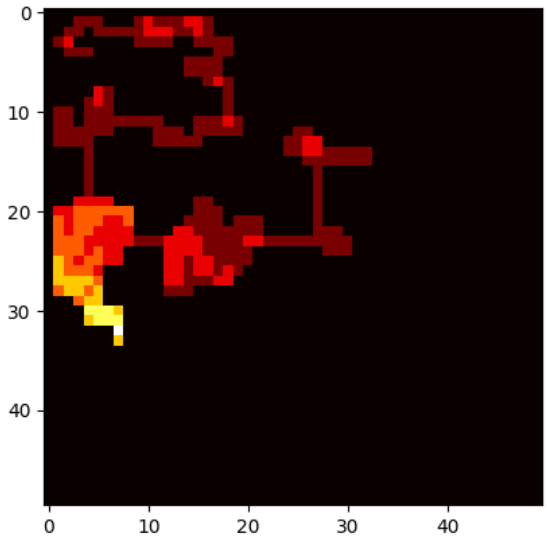
5.3. ábra. Agent 2



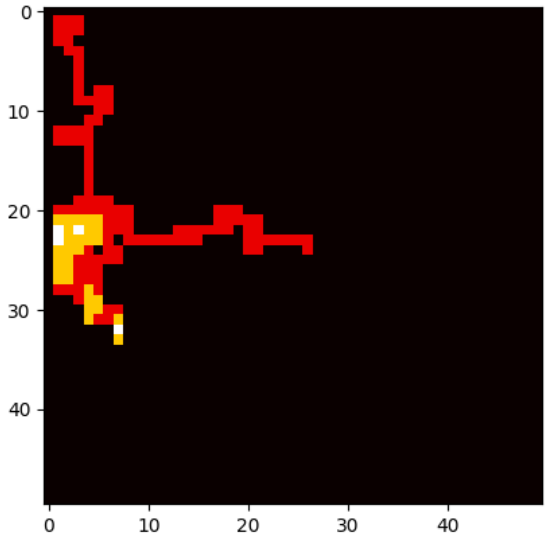
5.4. ábra. Agent 3



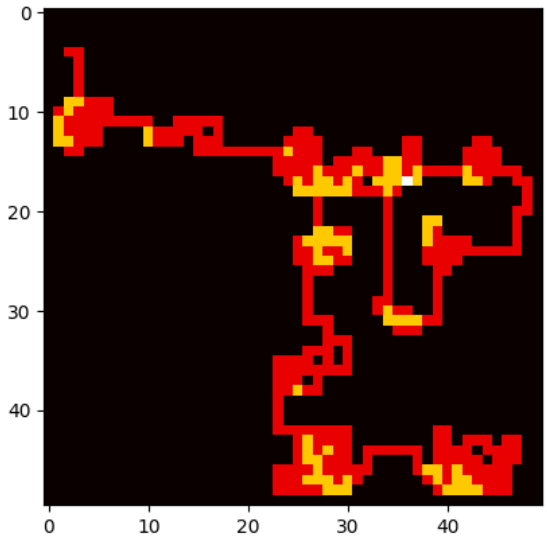
5.5. ábra. Agent 4



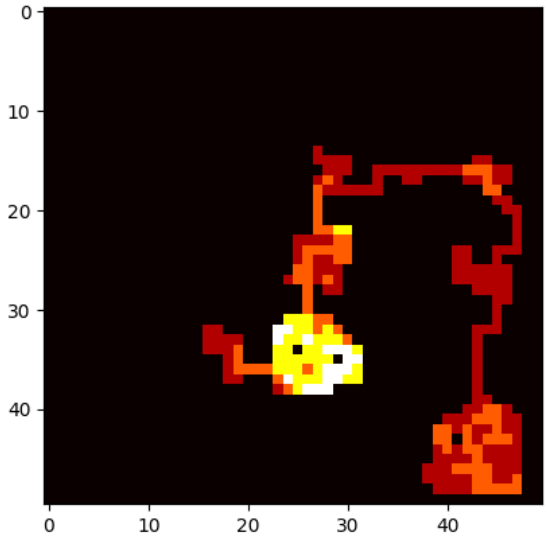
5.6. ábra. Agent 1



5.7. ábra. Agent 2



5.8. ábra. Agent 3



5.9. ábra. Agent 4

6. fejezet

Összefoglalás

A félévben a szakdolgozaton dolgozva sikerült felélénkíteni az egyetem alatt megtanult java ismereteim, és bővíteni azokat a java Swing-gel. Emellett nagy segítséget nyújtott a Szoftvertechológiák nevezetű tárgyon tanult ismeretek, amelyekben megtanultuk, hogy hogyan tervezzünk meg egy jövőben elkészülő alkalmazást. A félévben kisebb nehezségekbe ütköztem a tervezés és implementáció során, amelyek megoldása fejlesztette látásmódom, így ha legközelebb valamilyen hasonló feladatom lenne képes lennék hatékonyabban megoldani azt.

A tesztek során néhány ágens látványosan kevés utat tett meg a szimulációban, ennek oka az, ha összetalálkozott egy másik ellentétes ágenssel, akkor valamelyik ágens a számolások után elpusztult.

A szimuláció futási ideje nagyban függ attól, hogy mennyire segítjük az ágenseket azzal, hogy teszünk-e több kulcsot a pályára, mint amennyi ajtó van. Hiszen, ha ugyanannyi kulcs van a pályán, mint ajtó és feltételezzük, hogy olyan helyekre vannak eltéve, ahol nem lehetséges az, hogy úgy használják el minden kulcsukat, hogy ne férjenek hozzá további kulcsokhoz. Ebben az esetben annak az ágensnek, akinek elfogyott a potenciálisan megszerezhető kulcsok száma, akkor a másik ágens útvonalát kell megkeresni és követnie azt a tovább haladáshoz.

Irodalomjegyzék

[1] Ken Arnold, James Gosling, and David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.

[2] codeman38. Yoster Island Font. `https://www.1001fonts.com/yoster-island-font.html/`, 2022.

[3] Robert Eckstein, Marc Loy, and Dave Wood. *Java Swing*. O'Reilly and Associates, Inc., USA, 1998.

[4] Red Hook Studio. Darkest Dungeon. `https://www.darkestdungeon.com/`, 2022.

[5] watabou. Pixel Dungeon. `https://watabou.itch.io/pixel-dungeon`, 2022.

CD Használati útmutató

A CD az alábbi jegyzékeket tartalmazza:

- szakdolgozat: A szakdolgozat LaTeX forráskódját és PDF-t tartalmazza.
- Pela: Az alkalmazás jegyzékeit tartalmazza, a Jar file-t és néhány előző testet, amelyek a test1.txt és test2.txt.

A Pela jegyzék tartalma:

- bin jegyzék,
- src jegyzék: Java file-ok találhatóak meg az ebben talált jegyzékekben.
- res jegyzék: Használt képek, font-ok és txt az alkalmazásban.

A kód futtatása:

- a Jar file elinditásával lehetséges.
- A program a Java legújabb verziójában írodott, java 18.