

**Debreceni Egyetem**  
**Informatikai Kar**  
**Számítógéptudományi Tanszék**

**Mesterséges intelligencia valós idejű stratégiai játék  
implementációjában**

**Témavezető**

**Kádek Tamás**

**Tanársegéd**

**Készítette**

**Bari Sándor**

**Programtervező informatikus Bsc III. évf.**

**Bírálatra való beküldéséhez hozzájárulok:**

**Debrecen  
2014**

## **Köszönetnyilvánítás**

Köszönöm témavezetőmnek, Kádek Tamásnak a témavezetést és a dolgozat elkészítéséhez nyújtott segítséget.

# Tartalomjegyzék

|                                             |    |
|---------------------------------------------|----|
| 1. Bevezetés.....                           | 1  |
| 2. A játék szabályai.....                   | 5  |
| 2.1 A térkép.....                           | 6  |
| 2.2 A játékosok és a játék menete.....      | 7  |
| 2.3 A játék célja.....                      | 8  |
| 2.4 Épületek és egységek.....               | 9  |
| 2.5 Összegzés.....                          | 10 |
| 3. A fejlesztési környezet.....             | 11 |
| 4. Az alkalmazáslogika bemutatása.....      | 16 |
| 4.1 K-közép algoritmus.....                 | 16 |
| 4.1.1 Elméleti háttér.....                  | 16 |
| 4.1.2 Implementációs kihívások.....         | 19 |
| 4.2 Döntési háló.....                       | 22 |
| 4.2.1 Elméleti háttér.....                  | 22 |
| 4.2.2 Implementációs kihívások.....         | 27 |
| 4.3 Szabályalapú erőforrás-menedzsment..... | 30 |
| 4.4 A teljes rendszer működése.....         | 35 |
| 5. A további fejlesztések lehetőségei.....  | 39 |
| 6. Összefoglalás.....                       | 42 |
| 7. Irodalomjegyzék.....                     | 44 |
| 7.1 Nyomtatott irodalom.....                | 44 |
| 7.2 Online irodalom.....                    | 44 |

## 1. Bevezetés

A valós idejű stratégiai játékok a mesterséges intelligencia kutatás népszerű alterületét alkotják, mivel összetett viselkedés megvalósítását kívánják egy olyan szimulációs környezetben, melynek bonyolultsága tetszőlegesen skálázható. A játékmenet általános lefolyása szerint a játékosok bizonyos gazdasági döntéseket hoznak, ezáltal erőforrásokra tesznek szert, például a játéktéren található bányák kitermelésével nyersanyagokat szereznek, melyeket aztán felhasználhatnak katonai erejük fejlesztésére, egységek kiképzésére. A játék bizonyos cél eléréséig tart, vagy addig, míg az egyik játékos meg nem semmisítette a többit.

A játék jellegéből adódó kihívások, melyekkel az ágens létrehozása során szembekerülünk, megfelelnek azoknak a kihívásoknak, melyek általános problémaként jelentkeznek a racionális viselkedést megvalósítani képes mesterséges intelligenciára irányuló kutatások során. Gondolhatunk itt többek között a részleges, bizonytalan információk, feltételezések alapján történő döntéshozatalra, az ellenfél rendelkezésére álló haderők modellezésére vagy éppen az erőforrás-menedzsmentre [9]. Mindezek a jellemzők ideális tesztkörnyezetté teszik ezeket a játékokat a legújabb algoritmusok kipróbálására, valamint ellenkező irányból, a játékokban jól teljesítő algoritmusok általánosíthatók valós világbeli alkalmazásokban történő felhasználásra. Összefoglalva: a játékoknak ez az alműfaja egyedülálló a mesterséges intelligencia kutatásban vizsgált játékok között, mivel alapvető problémákat vet fel, értékét pedig tovább emeli, hogy nyitott, dinamikusan fejlődő területről van szó, melyre nem jellemzők végleges, definitív megoldások.

Dolgozatom célja egy olyan ágens megvalósítása, mely képes az előbbieknél megfelelő, általam alkotott szabályrendszerű játékot játszani. Minimális követelményként jelentkezik, hogy a szabályok betartásával észszerű döntéseket hozzon, meggyőző viselkedést produkáljon. Ideális esetben azonban képes jól játszani, esetleg legyőzni a humán játékost.

A követelmények megvalósításának kulcsa a megfelelő döntési mechanizmus

kidolgozása, amely elég rugalmas ahhoz, hogy alkalmazkodjon a folyamatosan változó játék-állapotokhoz. Lehetővé kell tennie a gép számára a dinamikus, gyors reagálást, valamint aktív játéktílus kialakítását, értve ezalatt azt, hogy a gépi játékos a reaktív, alapvetően passzív megoldások helyett önálló stratégia kidolgozásával, aktívan tör a győzelemre, mindezt úgy, hogy maga a döntési folyamat nem vehet túl sok időt igénybe, hiszen valós idejű rendszerről van szó. Következésképpen az egyes algoritmusok számításigénye, a számítások időtartama nem léphet át bizonyos korlátokat. Ebből a feltételből látható, hogy a minimax, negamax algoritmusok nem alkalmasak a probléma megoldására. Még ha találnánk is módot arra, hogyan változtassuk meg ezeket úgy, hogy a gépi játékos képes legyen meghozni döntéseit segítségükkel, a szükséges számítások hossza játszhatatlanná tenné a játékot. Ezért másik megközelítésre, más algoritmusokra van szükség.

Dolgozatomban először a játékszabályok és a játékmenet bemutatására kerül sor, hiszen ezek írják le azt a világot, ahol a gépi intelligenciának tevékenykednie kell. A játékot ketten játszhatják, alapesetben az implementált ágens méri össze tudását egy humán játékossal. A szabályok ismertetését a felhasznált fejlesztőkörnyezet bemutatása követi, majd rátérek a dolgozat gerincét alkotó algoritmus- és architektúra-elemzésre.

A létrehozott mechanizmus három alappilléren nyugszik: a játék aktuális helyzetének elemzése a – többek között az adatbányászat területén használt – k-közép algoritmus segítségével történik, katonai döntések döntési háló segítségével születnek, mely a Bayes-háló kiterjesztése hasznosságelméleti alapokon, végül gazdasági döntések meghozatalát az ágens a szükségletek alapján, célorientált módon végzi.

Elsőként a k-közép algoritmust ismertetem részletesen. Alapvetően egy adathalmaz osztályokba sorolását végzi, számtalan helyen alkalmazzák sikerrel, többek között üzleti alkalmazásokban, geostatisztikában, vizuális adatfeldolgozásban (számítógépes látás) és még sorolhatnánk. A döntési mechanizmus során a gépi játékos az aktuális katonai helyzet elemzésére fogja felhasználni: a térképet zónákra osztja, ezek a zónák fognak megfelelni az osztályoknak, a besorolásra váró adathalmazt pedig a saját és az ismert ellenséges katonák helyzete alkotja. A zónákra osztás a térkép kiemelten fontos pontjai alapján történik, gondolhatunk itt a saját és ellenséges bázis helyére, az elfoglalásra váró pontok koordinátáira, és hasonlókra. A csoportosítás

elvárt eredménye az, hogy a gépi játékos pontos képet kap arról, melyik zónában milyenek az erőviszonyok, ezáltal következtetni tud többek között bázisának fenyegetettségére, vagy éppen arra, hogy egyes pontok elfoglalásánál mekkora ellenállásba ütközne. Az általános algoritmust egyes pontokon módosítottam, hogy jobban illeszkedjen a problématerhez. Az alap és a módosított algoritmus összehasonlítása visszatérő eleme a résznek.

A katonai döntéshez felhasznált döntési háló leírása a következő lépés. A háló egy irányított aciklikus gráf, melynek csomópontjai három típusba tartozhatnak: lehetnek megfigyelt, pontos adatokat tartalmazó csomópontok, melyekre bizonyíték-csomópont néven fogok hivatkozni, lehetnek a bizonyítékok alapján valószínűségi értékeket előállító valószínűségi vagy véletlen csomópontok, végül a lehetséges akciók közül az elvégzendő kiválasztását végző döntési csomópont. A döntések rendelkeznek valamilyen számszerűsített hasznosság-értékkel, a hasznossági csomópont alkalmas célfüggvény maximalizálásával a leghasznosabb döntés kiválasztását végzi. A hasznosságértékeket súlyozzuk, mely a véletlen csomópont által számított valószínűség-érték segítségével történik. A csomópontok típusainak elnevezésében az angol terminológiát követem [4, 626-627.].

A harmadik alkotóelem egy szükséglet-alapú modul, mely gazdasági döntéseket hoz, a rendelkezésre álló nyersanyagok felhasználását, épületek építését, egységek kiképzését végzi. Működése szabályok alapján, célorientált módon történik. A katonai modul igényeket nyújthat be számára, melyeket aztán ez vagy közvetlenül teljesít, ha ezeket teljesíthetőnek ítéli, vagy felülbírálja, módosítja az aktuális gazdasági helyzet függvényében.

Mindhárom technikánál külön indoklom, miért éppen azt választottam az ágens megvalósításához, továbbá szakirodalmi példákkal illusztrálom egyrészt ezeknek korábbi felhasználási módjait stratégiai játékokban, másrészt ezzel párhuzamba állítva a saját megoldásaimban rejlő újdonságokat, melyek bizonyos esetekben a mechanizmus általam végzett módosításaiból, bizonyos esetekben a létező mechanizmus szokásostól eltérő módon történő alkalmazásából fakadnak. Természetesen az alfejezetekben a felmerülő implementációs kérdések, valamint ezek megoldásai is teret kaptak.

Az alegységek elméleti és implementációs leírása után külön fejezetet szentelek a döntési folyamat egységes, átfogó bemutatásának, melynek célja az almodulok integrációjának

ismertetése, valamint az ágens hatékonyságának elemzése, beleértve annak vizsgálatát, hogy a célkitűzésekből mi és hogyan teljesült. Fontosnak tartom, hogy a részek működése után a részek által alkotott egésről, a modulok együttműködéséről is képet adjak, mert ezáltal válik érthetővé, hogy az almodulok együttesen hogyan hozzák létre a racionális ágenst.

Az utolsó előtti fejezet a továbbfejlesztés lehetséges módjait, irányait írja le. A továbbfejlesztés történhet egyrészt architekturális, másrészt algoritmikus szempontból, továbbá a három nagy modul makro-szintje mellett kisebb mikro-szinteken is. Végül a dolgozatot összegzés, a megfelelő konklúziók levonása zárja.

## 2. A játék szabályai

A játék katonai szimulációs játék, amely valós időben játszható, azaz a játszma menete folyamatos, bármelyik játékos bármikor léphet. Ketten játsszák, egy humán játékos és gépi ellenfele. A játék alkotóelemei: a térkép, amely a játszma tere vagy "világa", a térképen található nyersanyaglelőhelyek és stratégiai pontok, valamint a játékosok rendelkezésére álló (katonai és egyéb) épületek, egységek.

A játék világa a hagyományos osztályozási szempontokat alapul véve [4, 40-46] csak részben megfigyelhető, mert a játékosok csak azokat a területeket látják, melyekre befolyásuk kiterjed, determinisztikus, mivel az állapotok egymásra következése csak a játékosok akcióitól függ, szekvenciális, hiszen a játék folyamatosan zajlik, valamint dinamikus és folyamatos, mindkét tulajdonság a valósidejűségből fakad, tehát a játékosok bármikor hozhatnak döntést, ezáltal a játékos döntési folyamata közben is megváltozhat a környezet, a játék folyamatosan zajlik, állapotai szabadon alakulnak egymásba.

Hasonló stílusú játékokat kereskedelmi forgalomban is találhatunk, a valós idejű stratégia a videojátékok népszerű műfaja. Jelen dolgozat és az egész projekt ihletője a StarCraft sorozat, mely három faj, az emberek, a rovarszerű zergék és a technológiailag rendkívül fejlett, rejtélyes protossok háborújáról szól<sup>1</sup>. További példaként említhetjük az Age of Empires sorozatot, amelyben a történelem különböző korszakainak háborúit játszhatjuk újra valamelyik nemzetet irányítva<sup>2</sup>.

A hasonló játékokban, mint az a fenti két példából is kiderül, általában több irányítható faj, nemzet közül is választhatunk, melyek mindegyikének megvan a maga erőssége és gyengéje. Ezzel együtt jár az is, hogy a játék tervezőinek nagy figyelmet kell fordítaniuk arra, hogy a játékmenet kiegyensúlyozott maradjon. Az általam fejlesztett játékban mindkét oldal azonos egységekkel rendelkezik, így ez a probléma nem merül fel, továbbá az azonos feltételek

---

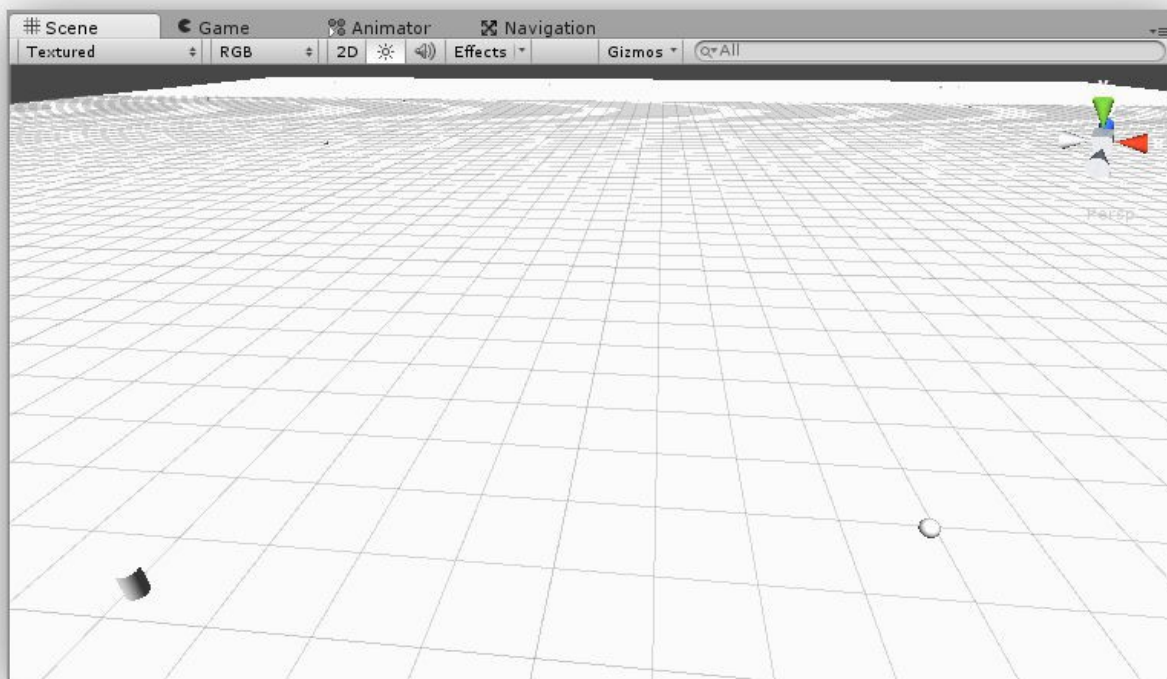
<sup>1</sup> <http://us.blizzard.com/en-us/games/sc/>

<sup>2</sup> <http://www.ageofempires.com/>

szükségesek ahhoz, hogy pontos képet kapjunk az ágens humán játékkal szembeni teljesítményéről.

A játék bemutatása a következő tagolást követi: először leírom a főbb tudnivalókat a térképről, majd ismertetem a játékosokra és a játékmenetre vonatkozó szabályokat. Ezután bemutatom a játék célját, azaz egy-egy játszma megnyerésének lehetőségeit. Végül rövid betekintést adok a játékosok által irányítható egységek jellemzőibe.

## 2.1 A térkép



Ábra 1: A kép a játékeret mutatja a szerkesztőprogramban, egységek nélkül. A bal alsó sarokban látható henger egy elfoglalandó stratégiai pontot, a jobb alsó sarokban látható gömb kitermelhető területet szimbolizál. A négyzethálós felosztás nem kap szerepet a játék során.

A térkép a játék tere, sík terület, melyen az egységek mozognak, és ahol a játékosok építkeznek. A terepnek van két speciális fajtája, melyek jelentősége kiemelkedő a játék

szempontjából: az egyik a nyersanyaglelőhely, az itt található nyersanyagokat a játékosok kitermelhetik, az így nyert tőkét felhasználhatják céljaik megvalósításához. A térképen található továbbá stratégiai szempontból fontos területek, a játék egyik célja ezek elfoglalása.

A játékosok a játék minden pillanatában látják a térkép egészét, vagyis ilyen értelemben a térképet nem kell felderíteni, viszont az ott bekövetkező valós idejű változásokat, például az ellenséges csapatok mozgását csak azokon a területeken látják, ahova kiterjed katonai befolyásuk. Minden egység, az épületeket is beleértve, rendelkezik a környezet érzékelésére szolgáló szenzorral, amely az egyszerűség kedvéért az adott egység fajtájától függő sugárral rendelkező gömb, melynek középpontjában maga az egység található. A gömb sugarán belül történő eseményeket az adott egység érzékeli. A humán és gépi játékosok egyaránt azokat és csak azokat az ellenséges manővereket látják a játék során, melyeket egységeik is látnak. Az így rendelkezésre álló töredékes információk alapján kell következtetniük arra, hogy mi történik a térkép egészén.

## ***2.2 A játékosok és a játék menete***

Mint már említettem, a játékot két fő játszhatja. Alapvetően egy emberi játékost és egy gépi ellenfelet feltételezek, ugyanakkor a játékszabályok megalkotásánál arra törekedtem, hogy azok alkalmazhatók legyenek több játékos esetén is.<sup>3</sup>

Egy játszma kezdetén a játékosok azonos feltételekkel rendelkeznek: bázissal a térkép egy-egy kijelölt pontján (kellő távolságra egymástól annak érdekében, hogy ne vesszen el a felfedezés öröme), induló tőkével, amely elegendő egy-két épület építéséhez, néhány egység kiképzéséhez, és kis létszámú induló haderővel.

A játékosok elsődleges célja a nyersanyaglelőhelyek felderítése, a nyersanyag kitermelése, majd az így nyert tőkéből épületek építése. Minden épület bizonyos funkciót lát el, illetve itt képezhetők ki a katonai egységek, ezek leírására később kerül sor. Az egységeket a játékosok a térképen kijelölt pontjára vezényelhetik, ezáltal megtámadhatnak ellenséges alakulatokat,

---

<sup>3</sup> Ez a törekvés az általános játékszabályokra és elvekre vonatkozik, az implementáció célja egy olyan játékprogram elkészítése, melyet egy ember játszik egy gépi ellenfél ellen.

biztosíthatják valamely terület védelmét, vagy éppen felderíthetik a térképen történeteket.<sup>4</sup>

Egy játszma általánosan tehát a következő forgatókönyvet követi: a játékosok felderítik a nyersanyaglelőhelyeket, megkezdik a kitermelést. Így lehetőségük nyílik épületek építésére. Az épületekkel már megkezdhetik katonáik kiképzését, saját haderejük fejlesztését. Ha elegendő katonát képeztek, megkezdhetik az előrenyomulást, azaz katonai befolyásuk kiterjesztését a térképen. Ennek eredményeként új területeket, nyersanyaglelőhelyeket szerezhetnek, elfoglalhatják a kijelölt pontokat, mindezt addig folytatva, míg valamelyik fél felül nem kerekedik a másikon.

### ***2.3 A játék célja***

Egy játszma valamelyik játékos győzelmével zárul. Győzelem elérése két módon lehetséges, a legegyszerűbb az ellenfél bázisának elpusztítása. A második lehetőség a térképen megjelölt kiemelt stratégia pontok mindegyikének elfoglalása (tehát a katonai befolyás kiterjesztése ezen területek egészére) és elfoglalva tartása bizonyos ideig. Ha ezen időtartam lejártáig a másik fél egyetlen területet sem tud elfoglalni, akkor a megszálló játékos megnyeri a játszmát. A játék kezdetén a játékosok nem ismerik a területek pontos helyét, azt a térkép felderítésével kell meghatározniuk, ismerik viszont az elfoglalandó pontok számát, illetve a későbbiekben természetesen azt, hogy ezek közül aktuálisan hányat tartanak befolyásuk alatt.

Döntetlen előfordulására van elvi lehetőség, ugyanakkor ennek esélye rendkívül csekély. Abban az esetben következhet csak be, ha a játékosok már az összes nyersanyagot kitermelték, de katonai ereje egyik félnek sem elegendő a játék lezárásához. A döntetlen helyzet kezelése: amint elfogy a nyersanyag a pályán, visszaszámlálás kezdődik. A játékból ekkor hátralévő idő rögzített, azaz játékállástól függetlenül mindig ugyanynyi, viszont bőségesen elegendő arra, hogy amennyiben valamelyik játékos képes a játék lezárására, megtegye azt. Ha ezen idő leteltéig nem születik döntés, a játszma döntetlennel zárul.

---

<sup>4</sup> Nyilván az egységek csak olyan helyre mozoghatnak, amely szabad, és ahol a terep járható.

## ***2.4 Épületek és egységek***

A játékosok által építhető épületek gyáarak, melyekben a játékosok által használható egységek képezhetők. Négy alapvető fajtáját különböztetjük meg: a gyalogos egységek kiképzésére szolgáló barakkot, a szárazföldi gépesített alakulatok létrehozására szolgáló, a légi egységeket termelő, végül a támogató egységeket előállító gyárat. Ez a felsorolás egyben képet ad a felhasználható egységek kategóriáiról is.

Mindegyik épület építésének van nyersanyagban kifejezett költsége. Természetesen az épületeknek területigényük is van, megfelelő méretű sík területen hozhatók létre. Az épületek fontos további jellemzője "életerejük", mely támadás hatására csökken. Ha ez az érték nullára zuhan, az épület elpusztul.

Az egységek közül legfontosabb talán a támogató egység. A harcban közvetlenül nem vesz részt, azonban képes a nyersanyag kitermelésére a lelőhelyen, valamint a kitermelt nyersanyag bázisra szállítására.<sup>5</sup>

A katonai egységek három csoportot alkotnak. Megkülönböztetünk gyalogos szárazföldi, gépesített szárazföldi és légi egységeket (azaz tankokat és repülőket). Mindegyik típus rendelkezik valamilyen fegyverzettel, melynek hatótávolsága és sebzése meghatározott, egy területtel, melyet szenzoraik érzékelnek, mozgási sebességgel, életerővel, amely ha elfogy, az egység elpusztul, végül az épületekhez hasonlóan nyersanyagban kifejezett képzési költséggel. Az egységekről elmondható, hogy a tankok erősek nagy tűzerővel, de lassúak és szenzoraik kis hatótávolságúak, a repülők gyorsak, viszonylag nagy (de a tankoknál kisebb) tűzerővel, ellenben költségesek, míg a gyalogosok bár gyenge egységnek számítanak, de költségük alacsony, így előállításuk nagy tömegben történhet.

Az adott egységek tüzet nyitnak a szenzoraik hatósugarába kerülő ellenséges egységekre, épületekre, ezáltal az egység típusának megfelelő mértékű sebzést okozva neki, így képesek ellenségeik elpusztítására. A model egyszerűsítése érdekében feltételezzük, hogy ha a célpont kiválasztása megtörtént, akkor a találat minden esetben sikeres, valamint az okozott sebzés konstans, azaz nem függ az egység típusán kívül egyéb tényezőktől. A harcot addig folytatják,

<sup>5</sup> Egységnyi nyersanyagot a játékos onnantól kezdve használhat fel, hogy az megérkezett a bázisára.

míg ellenséget látnak, el nem pusztulnak, vagy olyan parancsot nem kapnak, mely más tevékenység végzésére nem készíti őket.

Az első táblázat összefoglalja az egységek és épületek leglényegesebb tulajdonságait.

| <b>Egységek</b>         | <b>Ár</b> | <b>Életerő</b> | <b>Harci érték</b> | <b>Szenzor sugár</b> | <b>Sebesség</b> |
|-------------------------|-----------|----------------|--------------------|----------------------|-----------------|
| Gyalogos                | 15        | 30             | 5                  | 10                   | 5               |
| Támogató egység         | 20        | 80             | 0                  | 15                   | 7               |
| Tank                    | 35        | 100            | 15                 | 5                    | 3               |
| Repülő                  | 50        | 50             | 10                 | 10                   | 15              |
| Barakk                  | 100       | 300            | -                  | 30                   | -               |
| Támogató egységek gyára | 150       | 300            | -                  | 30                   | -               |
| Tankgyár                | 250       | 300            | -                  | 30                   | -               |
| Repülőgyár              | 500       | 300            | -                  | 30                   | -               |

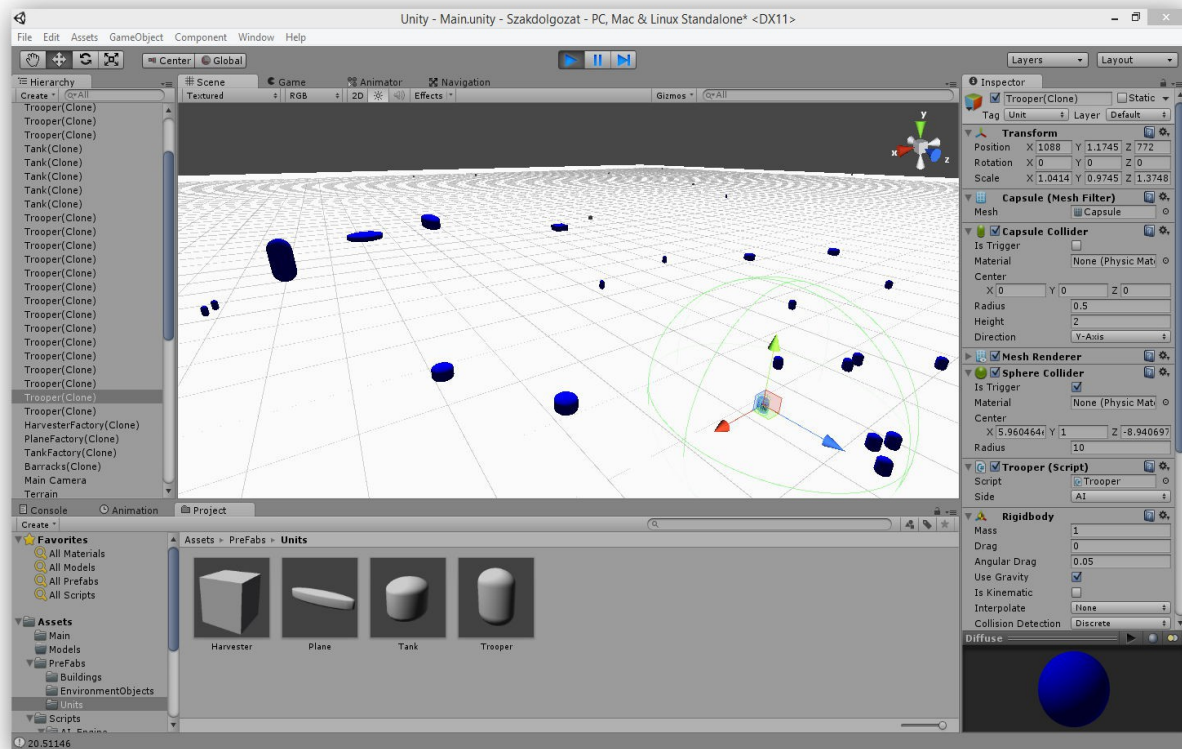
*Táblázat 1: Az egységek és épületek adatai*

## 2.5 Összegzés

A játék alkotóelemei tehát: az azt játszó két játékos, a térkép és ennek jellemzői, úgymint terepviszonyok, stratégiailag kiemelt pontok, nyersanyaglelőhelyek, valamint a játékosok rendelkezésére álló egységek és épületek. A felek arra törekednek, hogy megfelelő stratégia kidolgozásával, majd egységeiknek taktikai parancsok kiadásával legyőzzék ellenfelüket a katonai szimuláció során.

A dolgozat célja a fenti szabályoknak eleget tevő, azt játszani képes mesterséges intelligencia elméleti leírása és implementálása, valamint természetesen magának a játéknak a megvalósítása úgy, hogy az emberi játékos számára is kihívást jelentsen a gép legyőzése, így a játszmák érdekesek és élvezetesek legyenek.

### 3. A fejlesztési környezet



Ábra 2: A Unity felhasználói felülete.

A fejlesztési környezet kiválasztásakor legfontosabb szempontom az volt, hogy olyan környezetben végezhessem a munkát, amely kész eszközöket, egyszerűen kezelhető eljárásokat biztosít a mesterséges intelligenciával szorosan össze nem függő komponensek létrehozására, gondolok itt elsősorban a grafikára, felhasználói interfészre, továbbá elég szabadsággal kellett rendelkeznem ahhoz, hogy elképzeléseimet pontosan valósíthassam meg, a fejlesztőkörnyezet ebben ne korlátozzon. Mindezen szempontok alapján munkámat Unity motor segítségével végeztem, most röviden ezt szeretném bemutatni.

A Unity zárt forráskódú, kereskedelmi játékok fejlesztésére használt játékmotor, fejlesztői környezet, amely könnyen kezelhető grafikus felületet biztosít a játék komponenseinek létrehozásához, melyek viselkedését azután scriptek segítségével határozhatjuk meg. Három programozási nyelv kezelésére képes, a fejlesztő rendelkezésére áll a C# nyelv, természetesen a .NET teljes eszköztárával együtt, de mellette választhatjuk a JavaScriptet is, valamint a Boo nyelvet is, amely leginkább a Pythonra emlékeztet, attól eltekintve, hogy erősen típusos nyelvről van szó. Választásom a C# nyelvre esett, mivel a három közül ebben rendelkezem a legnagyobb fejlesztési tapasztalattal.

A környezetnek létezik ingyenesen használható és fizetős változata is. Az ingyenes változat –melyet én is használtam –tudása csaknem megegyezik a fizetős változatéval, bőven elegendő ahhoz, hogy érdekes, színvonalas játékot készítsünk segítségével. A fizetős változat ingyenesen nem elérhető funkciói olyan stúdiók számára lehetnek fontosak, amelyek kereskedelmi forgalomba szánt játékokat fejlesztenek. A környezetet nemcsak játékok fejlesztésére használják, hanem többek között épületek interaktív bejárását lehetővé tévő alkalmazások, oktatásra, specialisták kiképzésére szolgáló szoftverek is készülnek vele, elsősorban orvosi és katonai területeken. Nagy gyakorlatot igénylő vegyszeti és biológiai laborműveletek elsajátítását támogató virtuális környezettől kezdve kiemelt ügyfelek számára irodaházakat áruló ügynökségek termékbemutató szoftveréig alkalmazásával számtalan területen találkozhatunk<sup>6</sup>.

A környezet bemutatását két szempont alapján végzem: először a készen kapott, fejlesztő rendelkezésére bocsátott eszközökről ejtek néhány szót, ezek közül is azokról, melyeket felhasználtam a fejlesztés során, majd a scriptek általános működését ismertetem azért, hogy ezek a háttérinformációk segítsék a dolgozatom témáját képező program működésének jobb megértését [13].

A Unity grafikus felhasználói interfészt biztosít a program vizuális elemeinek létrehozásához. Importálhatunk kész modelleket, textúrákat és egyebeket, de ha ilyenek nem állnak rendelkezésünkre, akkor néhány beépített típus közül is válogathatunk. Egyszerű térbeli alakzatokról van szó, de lehetővé tették számomra, hogy könnyen megvalósíthassam a program

---

<sup>6</sup> További információk és letöltés: <http://unity3d.com/>

grafikáját, az mégis viszonylag látványos legyen. A gyalogosokat hengerek, tankokat korongok, támogató egységeket kockák, végül a repülőket lekerekített végű, talaj felett lebegő téglalapok szimbolizálják. Minden egységhez tartozik az adott egységet előállító gyár, ezek megjelenése megegyezik az egység megjelenésével, attól eltekintve, hogy néhány léptékkal nagyobbak. A stratégiailag kiemelt pontok jelképe a henger, a nyersanyaglelőhelyeket gömbök jelzik. Maga a játéktér egy terep-objektum, melyet beépített terepszerkesztő segítségével hozhatunk létre. Egyszerűen alakíthatunk ki domborzatot, megadhatjuk a legmélyebb és legmagasabb pont közti szintkülönbséget, a maximális meredekséget és még számtalan egyéb jellemzőt, ezek alapján pedig egészen realisztikus eredmény is születhet. Céljaim eléréséhez azonban ezúttal nem használtam fel ezeket az opciókat, megelégedtem az alapértelmezett síkszerű, négyzethálós tereppel.

A kettes ábrán látható a Unity felhasználói felülete. A terep közepén lent tankokat, jobbra gyalogos egységeket találhatunk. Az egyik gyalogos körül megjelenő félgömb az adott egység szenzorának hatósugara. Hátul épületek láthatók, balról jobbra haladva: barakk, tankgyár és repülőgépgyár. Ha a nagyobb interfészt tekintjük, lent láthatjuk az egységek modelljeit. Ezek a környezet által rendelkezésünkre bocsátott alapmodellek. Jobbra az aktuálisan kiválasztott gyalogos egységet mint játékobjektumot alkotó komponensek adatai sorakoznak, ezekről hamarosan szó lesz. Balra a játéktéren lévő összes objektum listája szerepel, alatta, a modellek mellett a projekt könyvtárszerkezete. Ebben a szerkezetben, a megfelelő alkönyvtárakban található a programkódok is, ezek szerkesztéséhez MonoDevelop szerkesztő áll rendelkezésünkre, melyet a Unity-val együtt kapunk.

Mint említettem, a Unity környezet a játékbeli objektumokat komponensekből álló egységeknek tekinti, melyek közül csak egyik az objektum megjelenített alakja. A vizuális formához rendelnünk kell egy Rigidbody nevű objektumot, melyet a beépített fizikai motor fog felhasználni számításaihoz. A Rigidbody tulajdonságai között olyanok szerepelnek, mint például súly, hat-e rá a gravitáció, stb. A Unity a fejlesztő közreműködése nélkül a megadott értékek és jellemzők alapján magától megvalósítja az alkalmazás fizikáját.

A Rigidbody mellett játékbeli objektumaim rendelkeznek egy gömb alakú Sphere collider komponenssel is. A collider vizuálisan nem jelenik meg, azonban ha egy másik objektum a

hatósugarába kerül, akkor lefut egy eseménykezelő metódus. A metódus törzse szabadon implementálható, ezáltal szabályozhatjuk, mi történjen ebben az esetben. A collider felel meg az egységek szenzorának, az implementált mesterséges intelligencia rendelkezésére tehát csak az egyes szenzorok által szolgáltatott információk összessége áll a döntéshozatal során.

A harmadik lényeges komponens maga a script, amelyet az adott objektumhoz rendelünk. Ennek segítségével forráskódból szabályozható az objektumok minden jellemzője, beleértve az eddig felsorolt komponensek tulajdonságait is. Minden script valójában egy osztályt tartalmaz, mégpedig azt, amelyik az adott játékbeli objektumnak a forráskód szintjén felel meg. Egyéb játékbeli objektumokat is egyszerűen elérhetünk a kódból, meghívhatjuk metódusaikat, üzeneteket küldhetünk nekik. Ezt címkézés segítségével érhetjük el: minden objektumot elláthatunk egy alkalmas címkével (a Unity nyelvén taggel), kódszinten pedig egy egyszerű metódushívással lekérhetjük a motortól az adott címkéjű objektumok listáját. Létrehozhatunk olyan osztályokat is természetesen, melyek nincsenek közvetlenül a játék valamelyik eleméhez rendelve, ezeket címkézni sem lehet, működésük, használatuk megegyezik az osztályok objektum-orientált fejlesztés során megszokott működésével.

A Unity az alkalmazás futása során végtelen ciklusban, frame-enként egyszer meghívja a játékobjektumokhoz rendelt script megfelelő metódusát, ezáltal frissítve az objektum állapotát. A metódus futása során nyilván hívhatunk további metódusokat is. Implementációs szinten a mesterséges intelligencia egy üres játék-objektum, mely a hozzárendelt osztályon kívül nem rendelkezik a korábban felsorolt komponensek egyikével sem – hiszen a játék során vizuálisan megjelennie nem kell. A fentebb ismertett mechanizmusnak köszönhetően a döntéshozatalt mozgásba lendítő metódus meghívásra kerülne frame-enként egyszer, ezt azonban nem tartottam célravezető megoldásnak. Ennek legfontosabb oka – mindamellett, hogy számításigényes eljárásokról volt szó – az, hogy nem eredményezne életszerű viselkedést, ha képkockánként születne új döntés. Egyrészt elveszne az emberszerű viselkedés illúziója, ugyanis a gépi játékos reakcióideje sokkal jobb lenne, mint humán ellenfeléé, másrészt a játék minősége is sokat romlana, hiszen a korábbi akcióknak nem lenne idejük kifutniuk magukat, a mesterséges intelligencia folyamatosan a döntéshozatallal lenne elfoglalva [1, 396-403]. Hogy ezt elkerüljem, egy időzítő segítségével szabályoztam, mikor szülessen új döntés. Általánosan jó eredményt

kapunk akkor, ha másodpercenként kétszer, legfeljebb háromszor engedjük a mechanizmus lefutását, ez az intervallum éppen megfelel egy átlagos emberi játékos reakcióidejének.

A fejezet zárásaként szeretnék néhány szót szólni az egységek által használt útvonalválasztási módszerről, mivel a Unity motornak köszönhetően ez is rendelkezésemre állt. Legelőször készítenem kellett a térképről egy navigációs mesh-t, mely alapvetően a terep járható és nem járható részeiről tartalmaz információt. A navigációs mesh előállítása beépített eszközök segítségével, egyszerűen elvégezhető. Ha ez elkészült, akkor minden egységhez, melyet mozgatni kívánunk, hozzá kell rendelnünk egy navigációs ügynök komponenst. Innentől kezdve nincs egyéb dolgunk, mint forráskód szinten beállítani annak a pontnak a koordinátáit, ahova vezényelni szeretnénk az adott egységet. A navigációs ügynök kezeli az útvonalválasztás problémáját a navigációs mesh adatait felhasználva. Ha olyan pontot jelölünk ki, mely nem járható, akkor automatikusan a legközelebb eső, még járható területre fog az egység navigálni. Az akadályok kikerülésének problémáját, az ütközések elkerülését szintén a navigációs ügynök kezeli.

A Unity eszköztárából az előbbieken bemutatottak azok, amelyeket felhasználtam az alkalmazás elkészítéséhez. A motor tudása ennél jóval nagyobb, azonban a program implementálásához, az algoritmusok teszteléséhez ennyi is elegendő volt. Kisebb problémát okozott, hogy a Unity alapvetően egy szálon futó alkalmazást tételez fel, viszont elég rugalmasnak bizonyult, így végül ez sem szabta gátját a fejlesztésnek. A Unity nagymértékben leegyszerűsítette a fejlesztési folyamatot, és lehetővé tette, hogy figyelmemet a mesterséges intelligencia megvalósításának elméleti és gyakorlati kérdéseire fókuszáljam.

## 4. Az alkalmazáslogika bemutatása

### 4.1 K-közép algoritmus

#### 4.1.1 Elméleti háttér

A k-közép algoritmus nagy múltra tekint vissza, első változatát 1967-ben dolgozta ki John MacQueen [11]. Az egyik legegyszerűbb, ezért legnépszerűbb adatosztályozási algoritmusról van szó, melynek elsősorban adatbányászati alkalmazásai ismertek.

Az adatosztályozási eljárás lényege az, hogy egy rendezetlen adathalmaz elemeit valamely tulajdonságuk szerint rendezve csoportokat alakítunk ki abban a reményben, hogy az így kapott mintázat magas információ-értékkel bír. A tulajdonság a k-közép algoritmus esetén egy  $n$  dimenziós vektor a vektortérben, az osztályozás alapját az osztályra jellemző átlagos vektor, és az osztályba tartozó vektorok távolsága szolgáltatja. Az algoritmus nagy előnye egyszerűsége, robusztussága, és az, hogy megbízható eredményt ad. Több, mint ezer publikáció jelent meg vele kapcsolatban, melyek nagyvonalakban két csoportba sorolhatók: az egyik elméleti kérdésekkel foglalkozik, míg a másik irányvonal a lehetséges alkalmazási területeket vizsgálja [14]. A következőkben bemutatom az algoritmus működését, majd rátérek néhány működéssel kapcsolatos problémára. A következő alfejezetben az algoritmuson általam végzett módosításokat ismertetem, különös tekintettel arra, hogy mi indokolta azokat.

A módszer algoritmikus bonyolultsága nem magas, négy lépésben történik az osztályozás. Először meg kell adnunk  $k$  darab kezdőértéket az osztályok középpontja számára,  $k$  meghatározza a kialakítandó osztályok számát. A lehető legjobb eredmény elérése érdekében érdemes az induló értékeket egymástól távol meghatározni. Miután megadtuk a kezdőértékeket, az algoritmus végigiterál az adathalmaz minden elemén, kiszámítva azok osztály-középpontoktól vett négyzetes euklideszi távolságát az alábbi függvény szerint:

$$D = \sum_{j=1}^k \sum_{i=1}^n \left( x_i^{(j)} - c_j^2 \right),$$

ahol  $x$  az aktuális adatelem, mellyel dolgozunk,  $c$  pedig az éppen vizsgált osztály középpontja<sup>7</sup>. A cél a függvény értékének minimalizálása, azaz adott  $x$  objektum abba az osztályba fog kerülni, melyhez a legközelebb esik [12].

Ha végeztünk az első iterációval, újra kell számolnunk az osztályok középpontjának helyét, az új középpont az osztályba tartozó elemek tulajdonság-vektorának átlaga lesz. Ezután újra végigiterálunk az adathalmazon, ismételten besorolva őket, az új osztályozás alapján megint módosítjuk a középpontokat, és így tovább, amíg valamilyen előre meghatározott konvergenciafeltétel nem teljesül. A leggyakrabban használt feltételek: előre meghatározott iterációs szám, az osztályok középpontja egy határértéknél kisebb mértékben változik csak, vagy az osztályok között nem történik tagcsere [14].

Az algoritmus működésével kapcsolatban azonosíthatunk két kritikus pontot: a távolságmérésre használt függvény és az osztályok száma nagymértékben befolyásolják a hatékony működést. Időnként felmerül az a probléma is, hogy az algoritmus csak olyan adatok osztályozására képes, melyek rendelkeznek valamilyen számszerűsíthető tulajdonsággal, ez pedig korlátozza felhasználhatóságát [14]. Azonban az általam implementált program esetében az osztályozás alapját helyvektorok képzik, ezért ez a kérdés nem volt releváns számomra. Következésképpen elegendő az előbb említett két kérdésre megnyugtató választ találnunk.

A  $k$ -közép algoritmus alapvetően érzékeny a zajra, kiugró értékekre [14]. A távolságmérésre használt függvény alapos kiválasztása azért is fontos, mert a kapott eredmény könnyen zajossá válhat, a nem megfelelően választott függvény torzíthatja a végeredményt. Az elemek egymástól vett távolsága alapvetően az adatok hasonlóságát fejezi ki, és mivel –ahogy arra hamarosan rátérek – a katonai döntések egyik pillére az általam implementált mesterséges intelligencia esetében az osztályozás végeredménye, ezért a pontatlanságok kiküszöbölése, a zaj csökkentése lényeges eleme a viselkedés hatékony megvalósításának. A már említett euklideszi távolság mellett használatos még a Manhattan-távolság és a Minkowski-távolság is. Abban a szerencsés helyzetben vagyunk azonban, hogy objektívan azonosítható a leghatékonyabb távolságmérő módszer, mely nem más, mint az alap algoritmusban is használt euklideszi távolság, így ebből a szempontból megbízhatunk az algoritmus működésében [5].

---

<sup>7</sup> A képlet pontos helye: [http://home.deib.polimi.it/matteucc/Clustering/tutorial\\_html/kmeans.html](http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html)

Érdekesebb kérdés az osztályok számának problémája, mivel az algoritmusnak ezt a priori módon tudnia kell, hogy elvégezhesse feladatát. A  $k$  értékének kiválasztása triviális esetben a felhasználó által kézzel megadott érték. Elméleti szempontból az algoritmus tesztelésére használt adathalmaz gyakran egyenletes eloszlás alapján működő generátorok segítségével jön létre. A generátorok száma határozza meg az osztályok számát. A probléma ezzel az, hogy valós problémák esetén, ahol nem számolhatunk generátorokkal, nem alkalmazható [15]. Hasonló problémák merülnek fel, ha statisztikai alapon próbáljuk meghatározni az osztályok számát, ugyanis arról nem áll rendelkezésünkre információ, hogy adataink milyen eloszlást követnek [15]. Visszaülva arra, hogy könnyen torzulhat az osztályozás eredménye,  $k$  értékének helyes megadása azért is fontos, mert az algoritmus működéséből következik, hogy nem lesz olyan adat, mely ne kerülne besorolásra valamelyik osztályba, ez pedig rossz osztály-szám esetében azt eredményezheti, hogy a létrejövő osztályok határai nagyon eltolódnak.

Az algoritmus mohóságával összefüggő probléma az üres osztályok problémája. Ha az iterációk lefutása után valamelyik osztály üresen marad, azt a szakirodalomban jellemzően hibaként kezelik, mely az osztályok számának hibás inicializálására vezethető vissza, és a szingularitási probléma nevet kapta. A szingularitási probléma kiküszöbölésére egy, az eredetivel szemantikailag azonos algoritmust hoztak létre, mely azzal megegyező teljesítményt nyújt anélkül azonban, hogy üres osztályokat hozna létre. [3].

Az utóbbi két pont, tehát a kiinduló osztály-szám, és az üres osztályok kérdésköre az, melyeknél algoritmusom eltér a normától. Az algoritmus működése, és az azzal kapcsolatos jellemző problémák áttekintése után a következő fejezetben azt mutatom be, hogyan alkalmaztam ezt a technikát, hogyan kezeltem az induló osztály-szám és az üres osztályok problémáját, ebből következően miben rejlik a  $k$ -közép általam létrehozott változatának újszerűsége.

#### **4.1.2 Implementációs kihívások**

Az implementált alkalmazás esetében a mesterséges intelligencia a saját és az ismert ellenséges egységeket sorolja osztályokba. Az osztályozás alapjául szolgáló tulajdonság az egységek helyzete, mely egy háromdimenziós vektor segítségével adott. Az osztályozás célja az, hogy a létrejövő besorolás alapján a térképet zónákra osztva információval rendelkezzen arról, hogy melyik területen milyen a saját és az ellenséges egységek aránya, ezáltal következtetéseket tudjon levonni olyan kérdésekre vonatkozóan, mint például melyik terület ellen tud a legnagyobb eséllyel sikeres támadást vezetni, mekkora veszély fenyegeti bázisát, és hasonlókat.

Az osztályok számának meghatározása a probléma jellegéből adódóan triviálisan adódott: minden ismert stratégiai pont egy osztály középpontja lett, továbbá a gép bázisa, a bázist alkotó épületek elhelyezkedéséből számolt átlagvektor alapján, illetve adott esetben, ha ismert, az ellenfél bázisa is egy osztályt alkot. Egyben ezek alapján történik a térkép zónákra tagolása is. A k-közép algoritmus futásának eredményeként a mesterséges intelligencia képet kap arról, hogy melyik zónában mekkora haderőt képviselnek saját egységei, és ott mekkora ellenállással kerülhet szembe. Ezek az információk adják a katonai döntéshozatal alapját, ugyanis a későbbiekben ismertetett döntési háló felhasználja majd a zóna-információkat.

Az üres osztályok nem problémát, éppen hasznos plusz információt jelentenek az osztályozás során. Ahogyan már említettem, jellemzően ezek létét hibaként értelmezik, alkalmazásomban azonban a mesterséges intelligencia ezek létét felhasználja a játék aktuális helyzetének elemzésére. Az üres osztály ugyanis számára azt az információt hordozza, hogy azon a területen nincs ellenfél, és ha egy elfoglalalendő pontról van szó, akkor oda könnyedén kiterjesztheti befolyását. A döntési hálónál bemutatásra kerül, hogyan számítja ki a gépi játékos az elfoglalás valószínűségét a haderők arányából, de már itt előljáróban megjegyezném, hogy az információ-érték abból fakad, hogy az üres osztályok elfoglalási valószínűsége a gép szempontjából 1, tehát –feltéve, hogy oda egységeket irányít –biztos eseményként kezeli. Alkalmazásom tehát a gyakorlatban demonstrálja a szingularitási probléma egy lehetséges megoldását –nevezetesen azt, hogy a jelenséget felhasználhatjuk a döntési mechanizmus finomítására.

Ahogyan az kiderült az algoritmus általános működéséből, minden egyég besorolásra

kerül valamelyik osztályba. Felmerülhet a kérdés, mi történik, ha a játék elején a gépi játékos még csak egyetlen osztályt ismer, saját bázisát? Az osztályozási mechanizmusból fakad, hogy minden egység, saját és ellenséges is ebbe az osztályba fog tartozni. De így akkor is, ha az ellenség éppen a térkép másik felén tartózkodik, arra a következtetésre juthatna a gép, hogy bázisát támadás fenyegeti. A szingularitási probléma kezelése mellett az algoritmuson végzett második módosításom az osztály-középpontok eltolódásának mértékét használja fel a döntési folyamat során. Külön tárolva a kiinduló értéket, kiszámíthatjuk a végső és a kezdő középpont közti távolságot. Ezáltal a gépi játékos rendelkezésére áll az az információ is, hogy mennyivel tolodott el az osztály. Mivel ezt számszerűsítettük, az eltolódás mértékével súlyozva az osztályozás eredményeként előállt információt egy újabb ponton finomíthatjuk az eredményből levonható következtetést. Az előbb említett példában hiába tartozik minden egység egyetlen osztályba, ha az ellenséges erők a bázistól távol koncentrálódnak, akkor a mesterséges intelligencia tudni fogja, hogy bázisát nem fenyegeti közvetlen veszély, és a védekezés helyett esetleg más akciót fog választani.

A stratégiai játékokban általában nem használnak a k-közép algoritmushoz hasonló adatbányászati osztályozó algoritmust. A döntési háló bemutatásánál még visszatérek rá, hogyan zajlik általában a döntéshozatali mechanizmus, ezen a ponton azt emelném ki, hogy a legjellemzőbb megoldás a játékban résztvevő felek befolyásának meghatározására a befolyási térkép elkészítése. Minden egység rendelkezik bizonyos harcértékkel, továbbá a térkép négyzethálós osztású. Ekkor az egység képes befolyást gyakorolni egy terület felett harcértékének és a terület távolságának függvényében. A befolyás csökkenhet lineárisan, exponenciálisan, ez a játék tervezőjének belátására van bízva. Így a térképet alkotó négyzetháló minden pontja esetén előáll, hogy arra mely egységek milyen mértékben gyakorolnak befolyást. Időnként használják még a grafikából ismert rekurzív elárasztásos megoldást is a befolyás kiszámítására, valamint a számítás gyorsítása érdekében esetlegesen egy területen az ott tartózkodó legerősebb egység befolyását veszik csak figyelembe [2].

A befolyás fentebb ismertetett módon történő meghatározás időigényes. Tovább nehezíti a helyzetet, hogy mivel nem veszi figyelembe, mit érzékel az adott egység aktuálisan, a térkép egészére elkészítve a térképet a gépi játékos abszolút tudással rendelkezik a játékállásról,

beleértve minden ellenséges egység helyzetét is. Ez pedig ellentmond a játékszabályoknak – az általam megfogalmazott részleges információkra támaszkodó döntéshozatal követelménye ugyanis általános a stratégiai játék esetében, és „fog-of-war” néven ismert. A játékosok csak azt a területet látják, amelyre befolyást gyakorolnak. Ha azonban a gép minden egység helyzetét ismeri, akkor képes csalni is. És valóban, a hasonló játékok esetén gyakran megesik, hogy a humán játékos a térkép egy eldugott zugában építeni kezd valamilyen különleges egységet, és annak ellenére, hogy gépi ellenfele nem tudhatna erről, mégis célzott támadást vezet oda. A számítás egyszerűsítéséből, azaz abból fakadó probléma pedig, hogy csak a legerősebb egység befolyását vesszük figyelembe, az a téves következtetés, hogy egy erős egység mindig legyőz sok gyenge egységet. Ez pedig koránt sincs így.

Az általam alkalmazott k-közép algoritmus számításigénye nem magasabb lényegesen a befolyási térkép készítésénél, és mivel csak az egységek szenzorai által szolgáltatott információra támaszkodik, ezért a csalás lehetősége ki van zárva. A számításigényt tovább csökkenti, hogy nem kell négyzetháló-alapú térképet használnunk, mivel a befolyást nem ilyen egységekben határozzuk meg.

A k-közép használatára találhatunk példát [6]. A szerző megemlíti a hagyományos módszer számításigényét, majd rátér a k-közép algoritmus hatékonyságára. A bázisok helyzetének meghatározására használja fel, ahol a térkép építésre alkalmas pontjai képezik az osztályokat, és a már meglévő valamint tervezett épületek helye a besorolásra váró adathalmaz. A cél az, hogy meghatározza a legbiztonságosabb pontot az építkezéshez, valamint a bázis közelítő helyzetét így kiszámítva támogassa az egységek útvonaltervezési mechanizmusát. A hatékonysággal kapcsolatban arra az eredményre jut, hogy a k-közép algoritmus nem bizonyult hatékonnak, mert a kapott eredmény információ-értéke alacsony, és az osztályok számának emelésén kívül nem tud egyéb megoldást a helyzet javítására.

A fentiekkel kapcsolatban megfigyelhetjük, hogy fontos, milyen probléma kezelését kíséreljük meg az algoritmussal. A bázis átlagos helyzetének meghatározása lehet releváns feladata a gépi játékosnak, de az egységek osztályozása sokkal több információval szolgál. A hivatkozott szerző megjegyzi, hogy az osztályozó algoritmust 5 másodpercenként futtatta az általa implementált mesterséges intelligencia. Ez érthető döntés, mivel egyrészt épületekről van

szó, és az építkezés ritka esemény, másrészt az osztályozás mellett külön számol befolyás-térképet is, és a két mechanizmus együttes számításgénye magas. Az általam elvégzett módosításoknak köszönhetően befolyás-térképre nem volt szükség, tovább az osztályozást aszinkron módon implementálva másodpercenként akár többször is elvégezhetővé vált a művelet.

Összefoglalva tehát a k-közép algoritmus kiegészítve az üres osztályok, valamint az osztály-középpontok kiinduló és végleges helyének nyilvántartásával hatékony eszköz arra, hogy a gépi játékos elemezze a játék aktuális helyzetét. A környezetből származó nyers információ így feldolgozva alkalmas arra, hogy támogassa egy döntési háló döntéshozatali mechanizmusát. A 4.4-es fejezetben bemutatom, hogy használja fel a háló az osztályozás eredményét, de előbb rátérnék a döntési háló elméleti hátterének ismertetésére, majd ezek alapján arra, hogyan születnek segítségével döntések.

## ***4.2 Döntési háló***

Amikor döntésről beszélünk a döntési háló kapcsán, akkor katonai döntésekről van szó. A gépi játékos akciói ugyanis két jól elkülöníthető részből állnak össze. Az egyik a szóban forgó háló által szolgáltatott katonai akció, a másik a következő alfejezetben bemutatásra kerülő szabályalapú modul által szolgáltatott gazdasági döntés. Jól indokolható, hogy miért van szükség két eltérő mechanizmusra, ugyanis egyszerűen arról van szó, hogy a katonai illetve gazdasági döntések problématerülete eltérő, és az adott problématerülethez leginkább illő eljárást kívántam alkalmazni. Mire mindkét modul bemutatásának a végére érek, reményeim szerint világos lesz, mik azok az eltérések, melyek az implementációs döntéseimet befolyásolták.

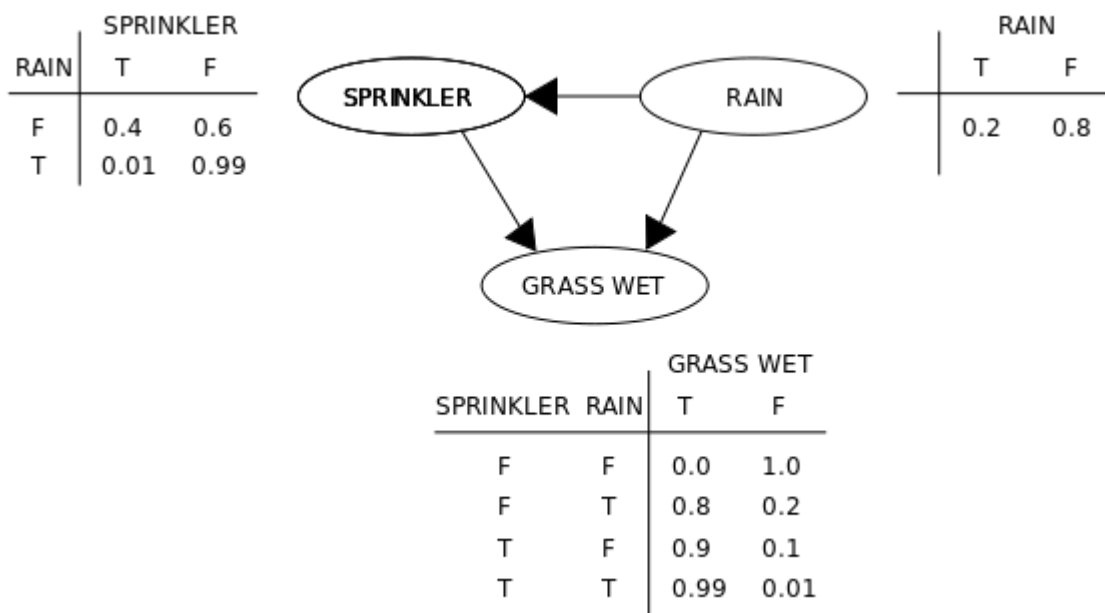
### ***4.2.1 Elméleti háttér***

A döntési háló nem más, mint a Bayes-háló kiterjesztése hasznosságelméleti alapokon. A definíció jelentésének megvilágításához először a Bayes-hálókat szeretném röviden bemutatni, majd rátérek arra, hogy pontosan mit jelent a hasznosság alapú kiterjesztés. A döntési hálók rendelkeznek szakirodalommal, azonban használatuk nem jellemző azoknak a problémáknak a megoldása kapcsán, ahogyan azt saját implementációmban tettem. Erre a kérdéskörre külön

kitérek az implementációs alfejezetben.

Az alapkérdés, melyre választ kellett találnom, az volt, hogyan tud a gépi játékost döntést hozni részleges és bizonytalan információk alapján. A Bayes-hálók éppen ilyen esetek kezelésére használhatók [6, 510-518].

A háló egy irányított gráf, melynek minden csomópontja megfeleltethető egy valószínűségi változónak, tehát minden csomópont egy eseményt jelképez, melyhez tartozik valamilyen valószínűségi eloszlás. A változó lehet diszkrét vagy folytonos, ez a problémától függ. A gráf nem tartalmazhat köröket, azaz irányított aciklikus gráfról van szó. Ha egy csomópontnak van gyermeke, akkor azt mondhatjuk, hogy a csomópont által reprezentált valószínűségi változó közvetlen befolyást gyakorol erre a gyermekre, ennek jelentése következésképpen az, hogy a szülő esemény bekövetkezte befolyásolja a gyermek csomópont által reprezentált esemény bekövetkeztét.



Ábra 3: A három csomópontot tartalmazó példa Bayes-háló.

A csomópontokhoz tartozik egy feltételes valószínűségi táblázat, mely megadja, hogy mekkora az esemény bekövetkeztének valószínűsége, feltéve, hogy a szülők bizonyos értékeket vettek fel. Úgyis mondhatnánk, hogy a táblázat egy lehetséges minivilág esetén mutatja az adott

csomópont által reprezentált esemény bekövetkezési valószínűségét [6, 512]. A háló tehát egy teljes feltételes valószínűségi eloszlást ábrázol, ez definiálja a háló szemantikáját [6, 513]. Ha kíváncsiak vagyunk egy esemény bekövetkeztének valószínűségére, akkor nincs más dolgunk, mint beállítani a háló paraméterváltozóit, a táblázatból kikeresni az egyes paraméterekhez tartozó valószínűségeket, végül ezek szorzata megadja a keresett esemény valószínűségét.

Egy példa segítségével szeretném megvilágítani a háló működését, ezt a hálót a harmadik ábra szemlélteti<sup>8</sup>. A probléma, melyre megoldást keresünk, a következő: mekkora a valószínűsége annak, hogy esik az eső, ha tudjuk, hogy a fű nedves? A csomópontok mellett látható táblázat a feltételes valószínűségi táblázat. Az eső csomóponthoz tartozó a legegyszerűbb, az eső valószínűsége mindössze 20%. A locsoló be- illetve kikapcsolt állapotát reprezentáló csomópont táblázata már kicsit bonyolultabb, azt ábrázolja, mekkora annak valószínűsége, hogy a locsoló be- illetve kikapcsolt állapotban van, feltéve, hogy esik az eső. Végül a nedves fű táblázata a legbonyolultabb, itt már mindkét változót figyelembe vesszük. Az eredmény kiszámításához vesszük az esik az eső – megy a locsoló – nedves a fű hármasszorzatát, hozzáadjuk az esik az eső – nem megy a locsoló – nedves a fű szorzatot (hiszen a locsolóról nem mondtunk semmit), majd ezt a teljes valószínűség tétele alapján elosztjuk – nem az összes lehetséges kombináció szorzatával, csak azokkal az esetekkel, amikor a nedves a fű változó igaz.

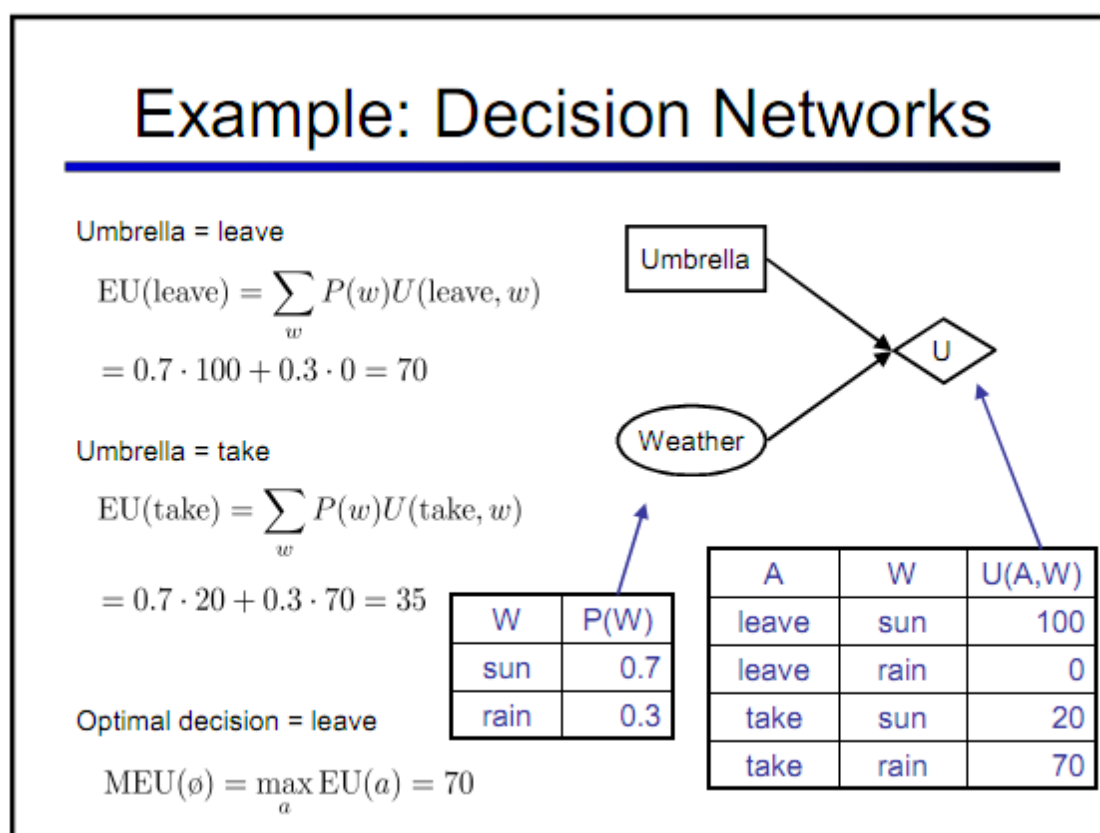
A példából jól látszik, hogyan történik egy bizonytalan állítás valószínűségének meghatározása. Ugyanakkor az eljárást számos kritika is érte, ezek elsősorban arra vonatkoznak, hogy a valószínűségi értékek és a változók számának meghatározása önkényes, másrészt a csomópontok közti kapcsolatok is némileg önkényesen kerülnek meghatározásra [17]. Ezek azonban számomra nem releváns kifogások, mivel a gépi játékos esetén már közelítő eredmények is képesek a racionalitás illúzióját kelteni. A Bayes-háló részletesebb bemutatását azért tartottam fontosnak, mert ennek alapján a döntési háló működésének megértése már egyszerű.

A döntési háló a fenti szerkezet kiterjesztése döntési csomópontokkal, valamint egy hasznossági csomóponttal [6, 626-628]. A döntési csomópontok azokat a pontokat jelképezik, ahol az ágens képes beavatkozni az események folyásába. Az ábra egy egyszerű helyzetet mutat:

---

<sup>8</sup> A kép és a példa forrása: [http://en.wikipedia.org/wiki/Bayesian\\_network](http://en.wikipedia.org/wiki/Bayesian_network)

hogyan dönt az ágens az esernyővel kapcsolatban az időjárás függvényében<sup>9</sup>? Az ovális csomópont a valószínűségi csomópont, a négyszögletes a döntési, ebben az esetben két lehetséges akcióval: vigyünk magunkkal az esernyőt vagy ne vigyünk magunkkal. A hasznossági csomópont a hasznosságokat táblázatos formában tartalmazza, mely azt mutatja, hogy attól függően, esik-e az eső, mennyire hasznos az esernyő. Látható, hogy leghasznosabb, ha jó az idő és nem visszük magunkkal az esernyőt, legrosszabb pedig, ha az esernyő nincs nálunk, viszont esik az eső. A számításhoz használt képlet ezután egyszerű: az adott akció – esemény páros hasznossága súlyozva az esemény bekövetkeztének valószínűségével.



Ábra 4: Példa a döntési háló mechanizmusára. [10] alapján.

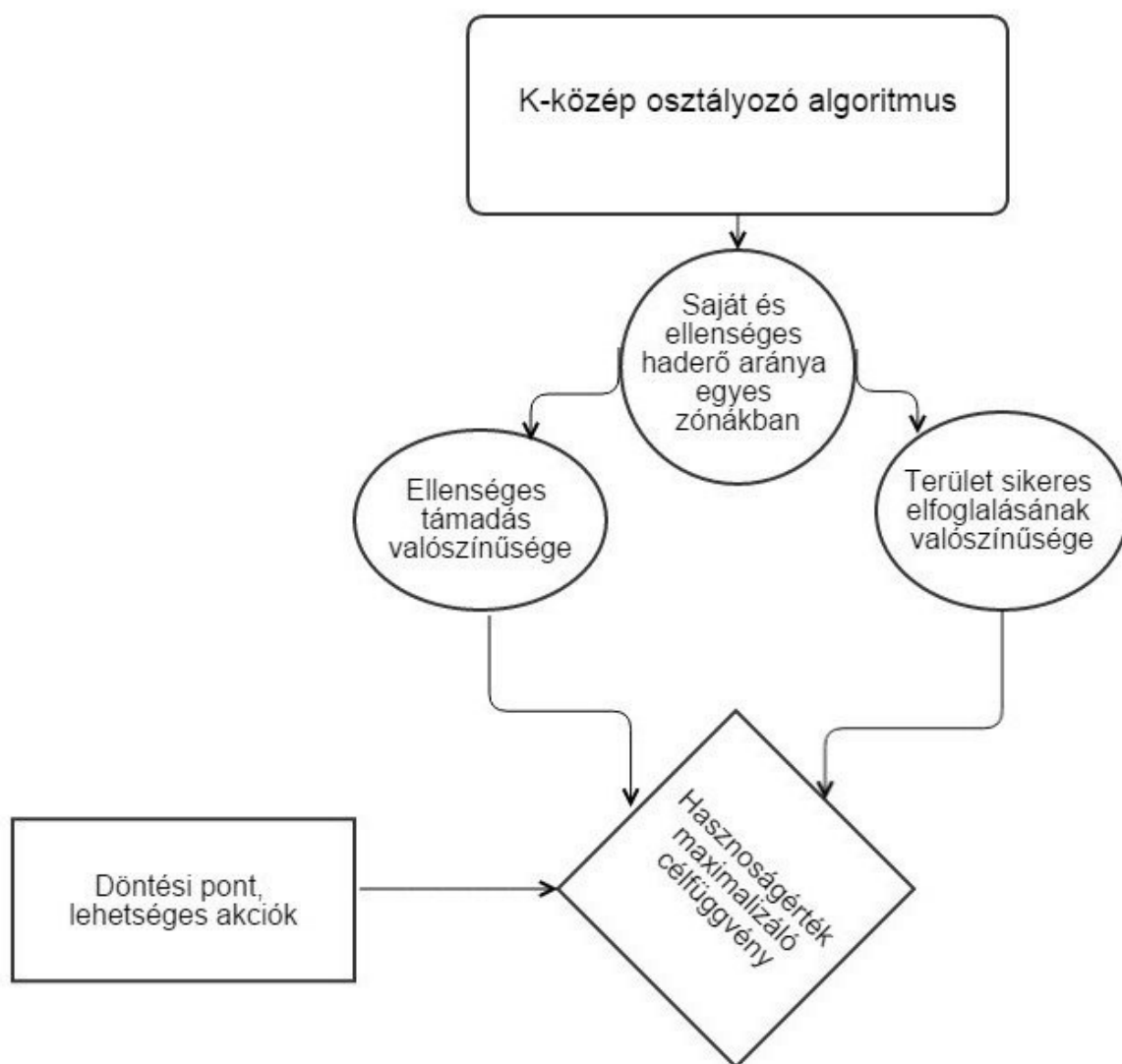
A hasznossági csomópont esemény a táblázatos forma természetesen nem kötelező, lehet alkalmas függvényt választva a hasznosság függvény maximalizálásra törekedni. A döntési háló

<sup>9</sup> Az ábra forrása: <http://inst.eecs.berkeley.edu/~cs188/fa10/slides/FA10%20cs188%20lecture%2018%20--%20decision%20diagrams%20%286PP%29.pdf>

megoldásának nevezzük a lehetőségek közül a ténylegesen elvégzendő akció kiválasztását. A művelet elvégzéséhez több algoritmus is rendelkezésünkre áll, egyik ilyen a változó eliminációs algoritmus, melynek alapja hasznosság függvény optimalizálása az adott esetben nem releváns csomópontok elhagyásával nagy döntési háló esetén [16]. Azonban az algoritmus számítási bonyolultsága magas, implementálása komplikált. Mivel a gépi játékosom által használt döntési háló nem tartalmaz túlzottan sok csomópontot, ezért a legegyszerűbb módszert választottam, a hasznosságok súlyozását a valószínűségekkel.

A döntési háló és működésének ismertetése után a következő pontban bemutatom, hogyan használja fel a hálót a mesterséges intelligencia döntések meghozatalára, és külön kitérek arra, pontosan hol kellett változtatnom az alapl működésen, hogy a döntési folyamat racionális döntéseket eredményezzen a játék folyamán.

#### 4.2.2 Implementációs kihívások



Ábra 5: A döntési háló szerkezete

A gépi játékos által használt döntési háló több pontban eltér az előzőekben ismertetett általános mechanizmustól. A döntéshozatal a k-közép osztályozás alapján történik. Az

osztályozás alapján meg tudjuk határozni annak valószínűségét, hogy a bázist támadás éri, mégpedig rendkívül egyszerűen:  $1 - x$ , ahol  $x$  a gépi játékos befolyása a bázis osztályában. Az elv alapvetően az, hogy minél magasabb a gépi játékos befolyása, a támadás valószínűsége annál kisebb, hiszen ez azt jelenti, hogy nem található nagyobb ellenséges haderő a bázis környékén. Az értéket még súlyozzuk az osztály-középpont elmozdulásának mértékével. Hasonló elvek alapján meg tudjuk határozni, hogy mekkora a legmagasabb esélye stratégiai pont elfoglalásának.

Az így kapott, 0 és 1 közé normált értékek lesznek a döntési háló valószínűségi csomópontjai által szolgáltatott értékek. A katonai akció alapvetően a fenti két érték alapján kerül kiválasztásra, tehát: azt, hogy a gépi játékos milyen akcióhoz folyamodik, az határozza meg, hogy mekkora valószínűséggel fenyegeti támadás bázisát, valamint az, hogy mekkora eséllyel tud stratégiailag jelentős pontot elfoglalni. A gép lehetséges akciói: bázisának megvédése, támadás, felderítés.

A döntési csomópontban egy adatszótárban találhatók a védekezés és támadás akciókhoz rendelt hasznosságértékek. A védekezés hasznossága magasabb, hiszen ha a gép elveszíti bázisát, akkor a játékot is elveszíti. A valószínűségértékekkel súlyozva a hasznosságértékeket megkapjuk az adott helyzetben leghasznosabb akciót.

Külön meg kell említenem a felderítés akciót, ugyanis az ehhez tartozó hasznosságérték kiszámítása speciális. A felderítés során a gép kiválaszt néhány gyalogos egységet, és a térkép véletlenszerű pontjaira irányítja őket. Majd újabb és újabb pontokat jelöl ki nekik, mindaddig, míg az egységeket az ellenség el nem pusztítja. A művelet célja az, hogy a térképet bejárva minél több információ álljon a gépi játékos rendelkezésére. Az akció jellege szerint olyan, hogy hasznossága nem ragadható meg kellőképpen valószínűségi változók segítségével. Míg azt látjuk, hogy az ellenséges támadás valószínűsége, a sikeres támadás valószínűsége egészen kézenfekvően meghatározza a hozzájuk tartozó akciók aktuális hasznosságát, addig ilyen változókat a felderítés esetén csak nehezen tudunk azonosítani.

A probléma megoldása szülte a döntési háló mechanizmusán végzett legjelentősebb változtatást. Ugyanis a felderítés akció nem függ valószínűségtől, hasznossága dinamikusan számítva áll elő. A gépi játékos tárolja, mikor indított utoljára egységet felderítésre. A művelet hasznosságértéke az eltelt idő függvényében lineárisan növekszik. Ebből következik, hogy eljön

egy pillanat, amikor a felderítés az elvégezhető leghasznosabb műveletté válik. Ekkor kiválasztja a mesterséges intelligencia az egységet, akit felderíteni küld, majd lenullázza a számlálót, így egy ideig nem küld ki újabb egységeket. Az idő függvényében változó hasznosságérték mutatja, hogy mennyire elavult az információ, mellyel dolgozik a gép. Ha tehát ez túllép egy tolerálható határt, akkor megkísérli frissíteni információit.

A stratégiai játékokban nem jellemző döntési hálók használata. Több eljárás is van, a két leggyakoribb a döntési fa és az állapotgép alkalmazása [5, 303-311].

A döntési fa könnyen implementálható mechanizmus, melynek lényege, hogy minden csomópontnál elvégzünk valamilyen tesztet. A játék aktuális állapota meghatározza a teszt eredményét, ezáltal azt, hogy a fa melyik ágán kell tovább haladnunk. Ezáltal végül megérkezünk egy levél csomóponthoz, amely egy elvégzendő akciót jelképez [5, 303-311].

Véges állapotgép esetén a gépi játékos előre rögzített állapotokban lehet, olyanokban, mint például “építkezés”, “támadás”, és aktuális állapotának megfelelő akciókat végez.[5, 318-341]. Feltételekkel szabályozhatjuk, hogy a játék állapotának milyen változása milyen állapotátmenetet váltson ki.

Mind a döntési fára, mind az állapotgépre igaz, hogy karbantartásuk nehéz, körülményes lehet, valamint minél bonyolultabb az implementálandó gépi játékos, annál nagyobbra nő méretük. Az állapotok, elágazások rendszere egy idő után kezelhetetlenül bonyolulttá válik. Természetesen léteznek módszerek ennek kezelésére, ilyen például a hierarchikus állapotgép [5, 327-340], ahol minden állapot rendelkezik saját belső állapotgéppel. A karbantartás bonyolultsága ebben az esetben is komoly problémát okozhat.

A döntési hálóknak (és Bayes-hálóknak) számos előnyük van a fentebb ismertetett két mechanizmussal szemben. A Bayes-hálók úgynevezett lokálisan strukturált rendszerek [6, 515]. Ez azt jelenti, hogy minden alkomponens csak korlátozott számú másik komponenssel lép kapcsolatba, így növekedési üteme az ilyen rendszereknek lineáris. A bonyolultságból fakadó karbantartási problémák tehát nem jelentkeznek ebben az esetben. A hálók alkalmasak tovább a bizonytalanság kezelésére is, a döntési fák és állapotgépek abból fakadóan, hogy tesztekre van szükségük, erre kevésbé alkalmasak.

Összefoglalva, a döntési hálók elméletéből kiindulva az általam implementált háló

mechanizmusának újdonsága egyrészt az, hogy egy adatosztályozási mechanizmus eredményéből származnak a felhasznált valószínűségértékek, másrészt az, hogy bizonyos hasznosságérték nem rögzített értékek súlyozásával, hanem dinamikusan kerülnek meghatározásra. Ezzel a két módosítással a döntési háló alkalmas eszközzé válik arra, hogy bizonytalan, részleges információk alapján következtetéseket vonjanak le a játék aktuális állapota alapján az elvégzendő akcióra vonatkozóan. A háló maga éppen a bizonytalansági tényező, és az egyes változók egymásra hatása miatt alkalmas struktúra. Az így kapott döntéshozatali mechanizmus hatékonyabb, de legalább olyan hatékony lehet, mint a játékfejlesztés során szokásosan használt egyéb eljárások.

A következő fejezetben a döntéshozatali mechanizmus újabb kiterjesztésére térek rá. Miután ugyanis a gépi játékos eldöntötte, mit szeretne csinálni, ezt a makro-szintű döntést alacsonyabb szintű műveletek sorozata követi. Itt arra kell gondolnunk, hogy parancsokat ad egységeinek, a játéktér megfelelő helyére vezényli őket. Ennek során azonban felmerülhet az a probléma, hogy szeretne valamilyen műveletet elvégezni, de nem áll rendelkezésére megfelelő számú katona, vagy ha rendelkezésére áll is, az akció erőforrásai jelentős részét igénybe veszi. Röviden: szüksége lehet új egységekre. A következő fejezetben bemutatom, hogyan zajlik az egységek kiképzése, mi indokolja, hogy a hálós szerkezetet elhagyva a gazdasági döntések esetében egy hagyományosabb, szabályalapú modult valósítottam meg, és ennek segítségével hogyan valósul meg egy szükségletalapú erőforráskezelési mechanizmus.

### ***4.3 Szabályalapú erőforrás-menedzsment***

Az erőforrások kezelése némileg más követelményeket támaszt a gépi játékkal szemben, mint a katonai döntések megozatala. Ha áttekintjük a stratégiai játékokkal kapcsolatos szakirodalmat, ritkán találunk konkrétan ezzel a kérdéssel foglalkozó forrásokat. Tegyük hozzá, ez nemcsak az erőforrások menedzselésére igaz, általánosságban is elmondható, hogy a szakirodalmi példák jellemzően megállnak az útvonalválasztó algoritmusok ismertetésénél, miközben legjobb esetben is mindössze valamilyen általános architektúrát ajánlanak a gépi játékos számára.

Visszatérve a katonai és erőforrás-kezelő modul közti különbségekre, a legszembevetőbb az, hogy míg a katonai döntéshozatal folyamán könnyen találhattunk olyan valószínűségi változókat, melyek befolyásolják a döntési mechanizmus végeredményét, addig a gazdasági döntéshozatal folyamán nem kézenfekvő ezeknek a változóknak az azonosítása. Míg ha az egyes akciók hasznosságáról általánosan beszélünk, annak van értelme, hiszen az egyes akciókhoz rendelt hasznosságértékek végsősoron meghatározzák a gépi játékos játéktípusát, addig nehéz általánosan beszélni az egységek hasznosságáról a kontextus ismerete nélkül. Fontos, hogy az egység hasznossága alatt nem harcértékét értem. Harcérték alapján egyértelműen eldönthető, hogy például a tank erősebb a gyalogosnál, de az nem igaz, hogy minden helyzetben tankot kellene gyártani gyalogos helyett. Következésképpen, míg a harcérték alkalmas mérőszáma az egység erejének, addig az egység hasznosságát más módokon kell meghatároznunk.

Létezik kísérlet arra, hogy az egységek kiképzésével kapcsolatban Bayes-háló segítségével döntsünk [7]. A vázolt döntési mechanizmus a következő: meghatározzuk az egyes típusokból a saját és ellenséges egységek arányát. Az arányokat 0 és 1 közé normálva valószínűségi eloszlásként felhasználva a Bayes-háló már ismertett működési elve segítségével meghatározzuk, hogy milyen egységre van szükség. Ezután azt, hogy ezt az egységet vagy termelő-egységet képezzünk-e ki helyette. Végül annak ismeretében, hogy milyen egységet akarunk előállítani, eldöntjük azt, hogy van-e az egység kiképzésére erőforrás, vagy előtte gyárat kell építenünk.

A fenti döntési folyamat több ponton is problémás. Kisebb probléma az, hogy először döntünk a katonai egység típusáról, majd arról, hogy ténylegesen ezt képezzük-e ki. Nagyobb probléma az, hogy a döntéshez ismernünk kell az ellenség haderejét, és az egész mechanizmus alapvetően reaktív jellegű, hiszen egy egység kiképzésének valószínűsége aszerint nő vagy csökken, hogy az ellenségnek mennyi van belőle. Azt hiszem, érezhető az is, hogy az egységek kiképzésével kapcsolatos fogalmak nem illeszkednek megfelelően a Bayes-háló szemantikájához. Erőltetett megoldásnak tűnik arányokat valószínűségi értéként felhasználni. A példával szemléltett ellenérvek játszottak szerepet abban, hogy a gazdasági döntések folyamán egy szabályalapú, pontosabban szükséglet-alapú rendszer alkalmazását találtam legcélravezetőbbnek.

A katonai döntés megszületése után a gépi játékos erőforrásokat különít el az akció

végrehajtásához. Három eset lehetséges: az első, hogy sikerül elegendő erőforrást allokálni a feladatra, és a művelet nem emészt fel erőforrásai túl nagy hányadát, a második, hogy a feladat végrehajtásához ugyan rendelkezésére áll elegendő erőforrás, de ezzel ki is merítette készleteit, végül megeshet, hogy nem rendelkezik a szükséges mennyiséggel. Erőforrás alatt most egységeket értek.

Az első eset a legkönnyebb, ugyanis a katonai döntésből nem születik gazdasági döntés igénye. A második és harmadik esetben azonban a katonai döntés végrehajtása csak akkor történhet meg, ha az együtt jár egységek kiképzésével is. A második esetben elegendő kevesebb egységet képezni, mintegy tartalékot képezni az adott típusból, a harmadik eset már több nyersanyag felhasználását kívánja.

A katonai döntéshozatalt végző modult almodulokra tagolhatjuk, mindegyik almodul, vagy mondhatjuk, almenedzser egy lehetséges akció végrehajtásáért felel. Az almenedzser feladata, hogy a rendelkezésre álló egységek alapján összeállítsa a felhasználni kívánt haderőt. Ha az almenedzser úgy találja, hogy a második vagy harmadik eset valamelyike áll fenn, akkor létrehoz egy üzenetet a gazdasági menedzser számára. Az üzenet tartalma, hogy melyik egységből mennyire van szüksége a katonai modulnak feladata ellátására.

A gazdasági menedzser fogadja az üzenetet. Az igény alapján eldönti, hogy elég nyersanyag és építési kapacitás áll-e rendelkezésére ahhoz, hogy teljesítse a kérést. Ha a válasz igen, akkor a megfelelő típusú egységet előállító gyárakat értesíti, hogy pontosan mennyi új egységre lenne a szükség. A gyárak pedig létrehozzák az új egységeket.

A gyártással kapcsolatban két probléma merülhet fel: nincs elegendő pénz az egység előállítására, valamint nincs elegendő gyártási kapacitás, hogy belátható időn belül eleget tegyenek a gyárak a kérésnek. A belátható idő ebben az esetben azt jelenti, hogy ha egy gyár gyártási kapacitását legalább kétszeresen meghaladja az igények száma, akkor azt a gyárat túlterheltnek tekintjük.

A gazdasági menedzser a gazdasági döntés meghozatala előtt a rendelkezésre álló összes nyersanyagból tartalékokat képez. Az egyik tartalékot épületek építésére, a másikat egységek gyártására lehet felhasználni, végül a harmadikat egyáltalán nem, azt raktározza későbbre, biztonsági okokból. Ha az egységek gyártására képzett tartalék elegendő arra, hogy legyártsa a

kért mennyiséget, akkor már csak a gyártósorok aktuális foglaltságát kell megvizsgálnia a gépi játékosnak. Ha nem áll rendelkezésére a szükséges nyersanyagmennyiség, akkor a rendelkezésére álló keretből az igényelt típusból a lehető legtöbbet próbálja meg legyártani. A több igényelt típus problémájára később még visszatérek.

Ha eldöntötte a mesterséges intelligencia, hogy hány egység gyártását engedi meg aktuális gazdasági helyzete, akkor megvizsgálja, hogy mekkora gyártási kapacitás áll rendelkezésére. Fentebb definiáltam a gyár túlterheltségének fogalmát. Ha a fenti definíció alapján úgy találja, nem áll rendelkezésére elegendő gyár, akkor az egység helyett épület építése mellett dönt. Ha rendelkezik elég gyárral, akkor szétosztja köztük a gyártandó mennyiségeket a gyár aktuális foglaltsága alapján, arra törekedve, hogy legkevesbé kihasznált kihasználttá váljon, míg a leginkább terhelt gyár ne váljon túlterheltté.

Az ismertett szükséglet-alapú megoldásnak számos előnye van a Bayes-hálóval szemben. Egyrészt a művelet logikája jobban illeszkedik a problématerre, a megoldás ahhoz hasonló, ahogyan természetesen gondolkodunk gyárak és termelés működéséről. Számítása is jóval egyszerűbb, és mivel mind a k-közép algoritmus mind a döntési háló aránylag számításigényes mechanizmus, ez fontos szempont.

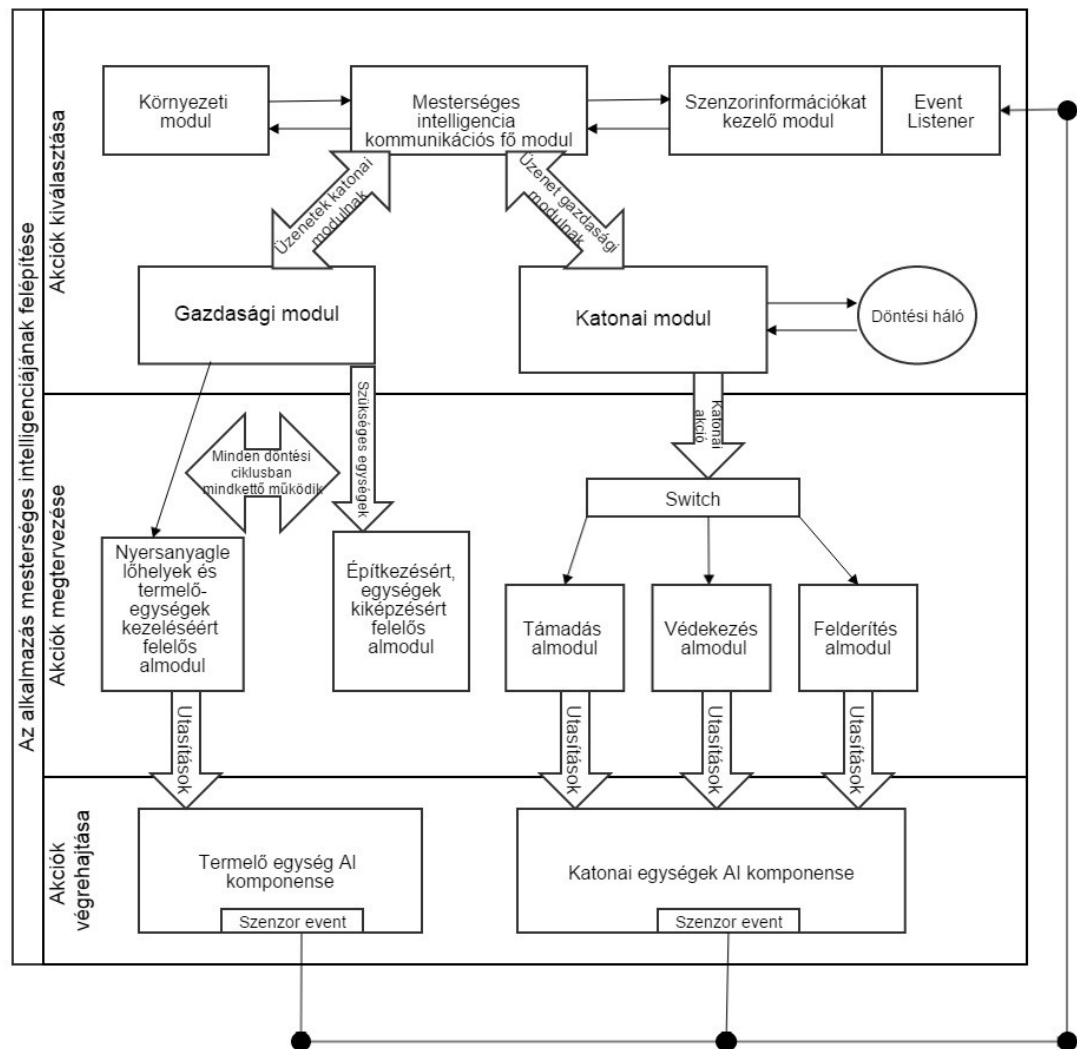
Legnagyobb előnye mégis az, hogy az eljárás közvetlenül illeszkedik a katonai döntéshozatali mechanizmushoz, kiszolgálja azt. A Bayes-háló alapján működő, reaktív megoldás helyett a szükségletalapú megközelítés, a modulok közti üzenetváltáson alapuló mechanizmus képes arra, hogy mindössze néhány szabály ellenőrzésével, betartásával megvalósítsa a gépi játékos igényeit. Az üzenetváltáson alapuló mechanizmusból következik a modulok izoláltsága, ez pedig azzal az előnnyel jár, hogy míg a katonai modul megvalósítása változhat, akár új technológiával kerülhet implementálásra, addig a gazdasági modul változtatására nincsen szükség, belső szerkezetének megtartásával alkalmazkodik bármilyen stílusú katonai döntéshozatali mechanizmushoz. Továbbá a szükséglet-alapú működés miatt a gazdasági döntéshozatal dinamikus, változatos viselkedést mutat, szemben egy esetleges kötöttebb megoldás merevségével.

Röviden szeretném bemutatni a nyersanyaglelőhelyek kezelését, mivel ez kulcsfontosságú a gépi játékos szempontjából. A nyersanyagok kinyerésének mechanizmusa egyszerű. A felderítés

során a gépi játékos megismeri, hogy hol találhat nyersanyagokat. Minden lelőhelyhez termelő egységeket rendel, arra törekedve, hogy az adott lelőhely kihasználtsága kellően magas legyen. A termelő egységek és lelőhelyek kezelését külön a megfelelő almenedzser végzi, ha nem áll a gépi játékos rendelkezésére megfelelő számú termelő egység, akkor a fentebb ismertett üzenetváltási mechanizmus segítségével igényelhet újabbakat. A nyersanyaglelőhely kihasználtságát egyszerűen az méri, hogy pontosan hány egység végzi a kitermelést. Ha ez a szám kellően magas, akkor a lelőhely megfelelően kihasználtnak tekintendő. Miután hozzárendelte az egységeket a lelőhelyekhez, ennek az almenedzsernek mindössze annyi dolga maradt, hogy ellenőrizze, nincs-e tétlen egység. Egy termelő egység tétlen lehet azért, mert éppen most gyártották le, és még nem kapott feladatot, illetve azért, mert kimerült az a lelőhely, amelynek kitermelését végezte. Az almenedzser a tétlen egységeket új lelőhelyekhez vezényli, ha vannak ilyenek. Az egység elmegy a lelőhelyig, elvégzi a kinyerési műveletet, majd visszaszállítja a kinyert nyersanyagot a bázisra.

Ezzel a mesterséges intelligencia mechanizmusának harmadik nagy egysége is bemutatásra került. Az egyes mechanizmusokat izoláltan megismerve nincs más hátra, mint a gépi játékos működésének egészét átfogóan, egy egységként bemutatni. A következő, egyben a fejezetet záró utolsó alfejezetben ezt teszem meg. Ez a lezárás – mindamellet, hogy nyilvánvaló célja a három nagy mechanizmus együttműködésének ismertetése, a teljes kép megmutatása –, a fejezetben elhangzottak összefoglalására is szolgál.

#### 4.4 A teljes rendszer működése



Ábra 6: Az implementált mesterséges intelligencia architektúrája.

A gépi játékos döntéshozatali mechanizmusának alapja három algoritmus: az adatosztályozásra szolgáló k-közép algoritmus, a döntési háló, amely alkalmas a játék természetéből fakadó bizonytalanság kezelésére, végül a szükséglet-alapon működő erőforráskezelő rendszer.

Az implementált mesterséges intelligencia két nagy modulból áll, a katonai és a gazdasági komponensből, mely kiegészül az egységek által szolgáltatott információkat kezelő szenzor-menedzserrel, és külön a környezeti információkat kezelő menedzserrel. Környezeti információk alatt az ellenséges épületek helyzetét, nyersanyaglelőhelyek, stratégiailag kiemelt jelentőségű pontok pozícióit értem. Röviden: a szenzorokat kezelő komponens a játéktér dinamikusabb, a környezeti menedzser a statikusabb információkat kezeli.

A katonai és a gazdasági modult további almodulokra bontjuk. A katonai modul almoduljai az egyes katonai akciók megvalósításáért felelnek, úgymint támadás, védekezés, felderítés. A döntési háló is a katonai modul részeként van megvalósítva, valamint a k-közép algoritmus is itt lép működésbe egy aszinkron hívás eredményeként. Az aszinkronitást az indokolja, hogy az osztályozás alapvetően erőforrásigényes művelet, ezért célszerű külön számban végezni. Miközben az algoritmus a munkáját végzi, a gazdasági modul elvégzi az erőforrások kezelését.

A gazdasági modul két alkomponenst tartalmaz, az épületek építését és az egységek kiképzését kezelő almenedzsert, valamint a nyersanyaglelőhelyeket és a termelő-egységeket felügyelő almenedzsert. Az aszinkron osztályozás futása közben ez a két almenedzser elvégzi feladatait. Ebből a megoldásból az is következik, hogy ha a teljes döntési folyamatot egy egységnek tekintjük, akkor a gazdasági menedzser mindig az előző katonai döntés alapján végzi munkáját. Ez a megoldás azért szükséges, mert katonai döntés nem születhet az osztályozás eredményének ismerete nélkül, ugyanakkor az osztályozással töltött időt nem szeretnénk elveszíteni. A gazdasági menedzser futása után az aszinkron osztályozást felügyelő eseménykezelő indítja a katonai döntéshozatal műveletét a k-közép algoritmus eredményei alapján.

Az osztályozás alapján meghatározható, hogy mekkora eséllyel várható támadás a humán játékos részéről, és milyen eséllyel képes az ágens stratégiailag kiemelt pontot elfoglalni.

Ezeknek az adatoknak a birtokában megtörténik a döntési háló kiértékelése az aktuális játékállás mellett, figyelembe véve a szenzor-információk elavultságának mértékét, melyet a felderítés művelet hasznosságának idő függvényében dinamikusan történő meghatározása fejez ki. Ezek alapján előáll az elvégzendő akció, az adott akcióhoz tartozó almodul végrehajtja azt.

Az akció végrehajtása az ágens jelenlegi fejlettségi fokán nem jelent túlzottan bonyolult műveletet. A gépi játékos pontosan tudja, hogy akciója melyik osztályra, azaz a térkép melyik zónájára vonatkozik. Rendelkezik azzal az információval is, hogy ebben a zónában milyen egységei vannak, ezek mekkora haderőt képviselnek, és ez a haderő hogyan aránylik az abban a zónában található ismert ellenséges haderőhöz. Ezek alapján mind támadás, mind védekezés esetében el tudja dönteni, hogy szüksége van-e további egységekre. Ha igen, először a szomszédos, majd az egyre távolabb eső zónákat vizsgálja meg, hogy honnan tud egységeket átvezérelni a célzónába. A zónában található, illetve ide vezérelt egységeknek kiadja a megfelelő parancsot: támadás esetén a zóna középpontja felé vezeti őket, védekezés esetén a bázis és az ellenség közös pozicionálja katonáit. Ha a művelet teljes haderejének túl nagy százalékát veszi igénybe, akkor üzenetet küld a gazdasági modulnak, hogy milyen egységekre lenne szüksége.

A felderítés annyiban zajlik másként, hogy a zónákon végigiterálva a gépi játékos kiválaszt néhány gyalogos egységet ott, ahol tud, majd kiadja nekik a felderítés parancsot. Az egységek rendelkeznek kezdetleges saját intelligenciával. Ha felderítés parancsot kapnak, akkor mindaddig, amíg ezt egy másik utasítás nem bírálja felül, kiválasztanak egy véletlenszerű célpozíciót a térképen, ha azt megközelítették, akkor másikat választanak, és így tovább. A felderítő egységek alapesetben addig végzik ezt, amíg el nem pusztulnak. A gépi játékos nem hívja vissza őket, inkább leírja veszteségként annak érdekében, hogy minél több információt szolgáltatassanak neki a játék állapotáról. Természetesen ha a felderítésre kiválasztott néhány gyalogos a teljes haderő jelentős százalékát képviseli, akkor erről is üzenetet kap a gazdasági menedzser. A gazdasági menedzser a kapott üzenetek alapján a következő döntési ciklusban elvégzi a szükséges műveleteket.

Röviden szeretném néhány mondatban bemutatni az egységek mesterséges intelligenciáját. Ez független a gépi játékostól, az egységek rendelkeznek bizonyos korlátozott mértékű önálló akarattal. A felderítés parancsra adott reakciójukat már bemutattam. Felderítés

mellett még megadható nekik egy pozíció, melyet el kell foglalniuk. Az egység mindent megtesz, hogy a megadott pozíció közelébe jusson. Ha szenzorának hatáskörébe ellenséges egység kerül, akkor azt megtámadja, kivéve a termelő egységek, ők katonai akciókban semmilyen szinten nem vesznek részt. Ha több ellenséges egységet is érzékel, akkor véletlenszerűen választ egyet. A modell egyszerűsítése érdekében feltételeztem, hogy a célpont kiválasztása után a támadás akció mindig sikerrel jár. A későbbi fejlesztések folyamán ez finomodhat, elképzelhető, hogy nem minden támadás lesz sikeres, vagy éppen az okozott sebzés mértékében szerepet kap a távolság is, ezek a lehetőségek nem szerepelnek a jelenlegi változatban. Az elvégzendő és elvégezhető fejlesztések a következő nagyobb fejezet témáját alkotják. Az egység tehát megtámadja az érzékelt ellenséget, típusának megfelelő mértékű sebzést okozva neki.

Az egységek mesterséges intelligenciájának felépítése azért előnyös, mert így a katonai akciókat felügyelő almenedzsernek nincs más dolga, mint utasítania az egységet a megfelelő paranccsal. A parancs végrehajtásának részleteit az egység saját hatáskörében elvégzi. Ezáltal az alkalmazásban jelenlévő mesterséges intelligencia rétegzetté válik, ahol egy nagyobb réteg a gépi játékos, mely két alrétegre bomlik, az általános műveletekről döntő nagyobb modulokra, és az ezek részét képező, a műveletek végrehajtását kezelő almodulokra. A másik réteg az egységek saját intelligenciája.

Az egyes mechanizmusok újdonságaira a korábbiakban már kitértem, azonban az architektúra egységként tekintve is rendelkezik előnyökkel. Volt már szó arról, hogyan képes a gépi játékos csalni. Ebben a felépítésben a döntéshozatalhoz elegendő az egységek szenzorai által szolgáltatott információ, az ágens nem tud csalni. A döntési háló jól kezeli a bizonytalanságot, a szükséglet-alapon szerveződő gazdasági modul pedig bármilyen játéktípust képes kiszolgálni.

A fentebb vázolt megvalósítással az ágens képes komplex viselkedés megvalósítására, racionális döntések meghozatalára. Ugyanakkor a felépítést és a mechanizmusokat számtalan ponton lehet még javítani, alakítani, valamint további eljárásokat bevonni az ágens hatékonyságának javítása érdekében. A következő fejezetben részletesen bemutatom az alkalmazás lehetséges fejlesztési, kiterjesztési pontjait, külön kiemelve azt, hogy mi indokolja az adott fejlesztést, és milyen eredményt, teljesítményjavulást várhatunk el annak eredményeként.

## 5. A további fejlesztések lehetőségei

Az ágens szerkezete számos ponton lehetőséget nyújt a további fejlesztésekre, új algoritmusok, ötletek kipróbálására. Röviden ezek közül szertnék bemutatni néhányat. A fejlesztés több szinten is elvégezhető, ezért a fejezet hármas tagolással rendelkezik: először magas szinten, a döntéshozattal kapcsolatos lehetőségekre térek ki, majd az almodulok szintjén vizsgálom a lehetőségeket, végül röviden az egységek saját mechanizmusának fejlesztéséről lesz szó. Ez a hármas tagolás pontosan megfelel az implementált ágens rétegeinek, így teljes képet kívánok adni arról, hogyan lehet a gépi játékos döntéshozatali mechanizmusában új technológiákat alkalmazni. Nem térek ki az alkalmazás egészére vonatkozó, illetve nem szigorúan véve mesterséges intelligencia algoritmusokkal kapcsolatos fejlesztési lehetőségekre. Értem ezalatt, hogy akár új egységek bevezetéséről, akár több játékos engedélyezéséről, ezzel kapcsolatban pedig hálózaton keresztül történő játék megvalósításáról nem esik szó ebben a fejezetben, miközben ezek is az alkalmazás fejlesztési tervei között szerepelnek.

Legmagasabb szinten, a döntéshozatali mechanizmussal kapcsolatban nyilvánvaló igényként merülhet fel a gépi tanulás kérdése. Izgalmas problémát vet fel egy olyan ágens implementálásának terve, amelyik képes új stratégiákat elsajátítani, stílusával alkalmazkodni ellenfelének játékstílusához.

A tanulással kapcsolatban lehetőség van egy jutalmazó rendszer bevezetése, amely a jó, sikeres akciókat jutalmazza, a sikerteleneket bünteti. Az így létrejövő ágens struktúrája többé-kevésbé megegyezne a Russel és Norvig által ismertetett általános struktúrák közül egy hasznosság alapú, tanuló ágenssel [6, 53-55]. A jutalmazási rendszer bevezetése lehet a jövőbeni fejlesztések egyik nagy iránya, azonban mivel még csak elgondolás szintjén létezik, ezért részletesebb megvalósítására nem térek ki.

A másik, tanulással összefüggő lehetőség annak vizsgálata, hogyan alkalmazhatók genetikai algoritmusok a döntéshozatali folyamat során. Az erre irányuló kutatások kettős szerepet tulajdonítanak a genetikai algoritmusnak [8]. Az egyik a stratégia megalkotása egyre finomodó, egyre hatékonyabb módszerekkel. Ez alkalmazható megközelítés az implementált

ágens esetén, ugyanakkor együtt járna a döntési háló mechanizmusának elvetésével. A másik szerep, amit a genetikus algoritmus kaphat, az nem más, mint egy, a terveket súlyozó függvény megvalósítása, amely segít az alkalmazandó terv, tevékenység kiválasztásában.

A döntéshozatallal kapcsolatban tehát a tanulás a fejlesztés legfontosabb iránya, az almodulokkal kapcsolatban az akciók végrehajtásának módján lehet sokat finomítani. A cél-orientált ágensekkel kapcsolatos kutatások támogatják a szükséglet-alapon szerveződő erőforrás-menedzsert. A katonai akciók tervezése is lényeges kérdés lehet. A játék világának további bonyolításával felmerül az igény az akciók részletes kidolgozására. Olyan lehetőségekre gondolok, mint terepviszonyok figyelembe vétele, támadási, védekezési stratégia kidolgozása. A már létező modulok információiból kiindulva ezek megvalósítható viselkedések az almodulok szintjén.

Visszatérve az erőforrások beosztására, érdekes kérdést vet fel, ha több különböző egységre nyújt be igényt a katonai modul, de nincs elegendő erőforrás az igény kielégítésére. Hogyan kell döntenie a gépi játékosnak, milyen egységeket képezzen? Nem biztos, hogy a legnyilvánvalóbb válasz, azaz a legerősebb egységből kell a legtöbbet képeznie, egyben a jó válasz is, hiszen ezek a legdrágábbak. A kérdést talán akkor tudjuk leghatékonyabban megválaszolni, ha dinamikus programozási problémaként, a hátizsák-probléma egyik speciális változatának tekintjük. A 0-1 hátizsák-probléma általánosítható arra az esetre, ha korlátlan mennyiség áll rendelkezésünkre az egyes tárgyakból. A gépi játékos esetén az összeállítandó hátizsák hasznossága felel meg a gyártott egységek együttes haderejének, míg a súlykorlát a gépi játékos egységek képzésére elkülönített tartaléka.

Végül egy kérdést szeretnék felvetni az egységek mesterséges intelligenciája kapcsán, ez pedig a térkép felderítésének problémája. A jelenlegi változatban, ahogyan arról már volt szó, véletlenszerű pontokat választunk a térképen, azonban ennek hatékonysága erősen kérdéses. A cél az, hogy olyan útvonalat jelöljünk ki a térképen a felderítők számára, melynek bejárása a lehető legnagyobb információértékkel bír. Ráadásul, mivel több felderítő-egységünk is lehet, nem elég egy ilyen útvonal azonosítása, hanem több útvonalat is választanunk kell, még ha akár csökkenő hasznossággal bírnak is. Az egység pályájának meghatározása naiv megközelítés szerint a görbék világába vezet, hiszen például valamilyen térkitöltő görbe alapján történő útvonalmeghatározás

esetén az egység kellő idő elteltével a térkép egészét bejárna – nyilván információ-tartalom szempontjából ez a legideálisabb eset. A görbék mellett esetleg alkalmas heurisztikus függvény használata is felmerülhet, ahol az útvonalválasztás az elvárt információ-tartalom alapján történik. Hangsúlyozom, hogy ezek nagyon általános, naiv felvetések, melyek túlmutatnak jelen dolgozat keretein, ugyanakkor a fejlesztés egy pontján kielégítő válaszra van szükség ezekkel kapcsolatban is.

A fejezetben bemutatam néhány lehetséges irányt, amerre az ágenst tovább lehet fejleszteni, javítani lehet. Ezek önmagukban is érdekesek, alapjául szolgálhatnak további kutatásoknak, ráadásul sokrétű és izgalmas kérdéseket feszegetnek, miközben közel sem fedik le az összes lehetőséget. Nem volt szó például a mesterséges intelligencia kutatásban olyan nagy múltra visszatekintő területről, mint a Fuzzy-logika. A következőkben a dolgozat egészének lezárásaként összefoglalom az elhangzottakat, különös tekintettel ezek újdonságaira a valós idejű stratégiai játékokkal kapcsolatos fejlesztésének kapcsán.

## 6. Összefoglalás

A dolgozat célja az volt, hogy egy katonai szimulációs játékot játszani képes, összetett viselkedésre képes gépi játékos megvalósítását mutassa be. Minimális célkitűzésként adódott, hogy az ágens többé-kevésbé racionális döntéseket hozzon, meggyőzően, a szabályok betartásával játssza a játékot. Ideális esetben képes jól játszani, megverni az emberi játékost csalás nélkül.

Az ágens megvalósításának alapja három algoritmus. Mindhárom algoritmus esetén módosítani kellett az eredeti működési mechanizmuson, vagy új elemeket kellett bevezetni, hogy jobban illeszkedjenek a problémátérhez.

Az adatok osztályozására szolgáló k-közép algoritmus által előálló információk adják a katonai döntéshozatal alapját. Így a térkép zónáiról, az ott található katonai viszonyokról pontos információt kap a gépi játékos. Ezeket az információkat egy döntési háló használja fel a katonai döntések meghozatalára. A háló a Bayes-hálók kiterjesztése egy döntési csomóponttal, mely azokat a pontokat jelképezi, ahol az ágensnek lehetősége van beavatkozni az események folyásába, valamint egy hasznossági csomóponttal, mely adott célfüggvényt maximalizálva kiválasztja a játék aktuális állása szerinti leghasznosabb tevékenységet. A választott tevékenység végrehajtása során új egységekre vonatkozó igények merülhetnek fel. Ezeket az igényeket a szükséglet-alapú erőforrás-menedzselő modul elégíti ki, továbbá ez felügyeli az épületek építését, a nyersanyaglelőhelyek kellő kihasználtságát, és a termelő-egységek feladatait.

A következőkben tételesen sorolom fel a fentiek alapján azokat az új elemeket, melyekkel nem találkozhatunk más stratégiai játékokban, vagy ha elő is fordulnak, az implementált gépi játékos azokat máshogy, másra használja fel.

- A k-közép algoritmus kapcsán újdonság alapvetően alkalmazása, mely a hagyományos befolyási térkép számítását váltja ki.
- A megszokott alkalmazások során a létrejövő üres osztályok jellemzően hibát jelentenek, itt a gépi játékos információ-értéket tulajdonít nekik, segítségükkel finomodik a döntéshozatali mechanizmus.

- Hagyományosan nem tartják számon az osztályok középpontjának változását. Az implementált ágens döntéseit tovább finomítja az induló és végső állapot közti eltérés.
- A döntési háló alkalmazása is újdonságnak, valós idejű stratégiai játékokban nem megszokottnak tekinthető.
- Hasznosságértékek kontextus-függő, dinamikus számítása is újdonság, mely további vizsgálatra szorul, hiszen a megfelelően választott eljárás nagymértékben hozzájárulhat a gépi játékos játékstílusának kialakításához.
- A döntési háló adatosztályozási algoritmus eredményét használja fel döntéshozatalhoz, erre korábbi példát nem találtam.
- Ahogyan nem ismert előttem szükséglet-alapú erőforrás-menedzselő modul sem, melynek döntései a katonai modullal történő üzenetváltás alapján születnek.

A felhasznált technikák az elvégzett módosításokkal alkalmassá váltak a minimálisan kitűzött cél elérésére, azaz elfogadható számításigény mellett a gépi játékos racionálisan méri fel a játék állását, hoz döntéseket.

Az ambiciózusabb követelmény elérése jelenleg kérdéses, az architektúra további fejlesztését, tesztelését igényli. A fejlesztéssel kapcsolatban bemutattam számos lehetséges irányt az alkalmazás több szintjén is.

Ezek közül legfontosabb az ágens tanulásával kapcsolatos, az önfejlesztő, új stratégiák dinamikus elsajátítására képes gépi játékos az igazi célkitűzés. Ha ez megvalósul, abból már következik a korábbi, jól játszó mesterséges intelligenciára vonatkozó célkitűzés megvalósulása is.

A tanulással kapcsolatos lehetőségek egyike egy jutalmazási rendszer kidolgozása és bevezetése. Másik érdekes irány genetikusan algoritmusok alkalmazása a döntési háló döntési mechanizmusának támogatására.

A tanulás mellett azonban egyéb fejlesztési irányokat is bemutattam, melyek segítségével lehetővé válik számos, a mesterséges intelligencia teürlétén jelenleg is aktívan vizsgált probléma kutatása. Úgy gondolom az, hogy a fejlesztést több eltérő, aktuális és izgalmas irányban lehet folytatni, önmagában jól mutatja a projekt koncepciójának igazi erejét.

## 7. Irodalomjegyzék

### 7.1 Nyomtatott irodalom

- [1] **Mark, D. (2009)** *Behavioral Mathematics for Game AI*. Charles River Media, Boston
- [2] **Millington, I. (2006)** *Artificial Intelligence For Games*. Elsevier, San Francisco
- [3] **Pakhira, M. K. A** Modified k-means Algorithm to Avoid Empty Clusters. *International Journal of Recent Trends in Engineering*, Vol 1, No. 1, 2009. 220-226.
- [4] **Russel, S - Norvig, P. (2009)** *Artificial Intelligence. A modern approach*. Prentice Hall, New Jersey
- [5] **Singh, A. -Yadav, A. - Rana, A.** K-means with Three different Distance Metrics. *International Journal of Computer Applications*, Vol. 67. Num. 10. , 2013 13-17.
- [6] **Stene, S. B. (2006)** *Artificial Intelligence Techniques in Real-Time Strategy Games – Architecture and Combat Behavior*. Msc Thesis, Norwegian University of Science and Technology, Department of Computer and Information Science. Trondheim, Norvégia
- [7] **Walther A. (2006)** *AI for real-time strategy games. Development of an AI for a real-time strategy game capable of learning new strategies*. Msc Diplomamunka. IT-University of Copenhagen, Koppenhága, Dánia

### 7.2 Online irodalom

- [8] **Aha D.W. - Molineaux M. - Ponsen M. (2006)** *Learning to Win: Case Based Plan selection in a Real-Time Strategy Game*. Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington DC, USA  
<http://www.cc.gatech.edu/~isbell/reading/papers/l2win.pdf>  
[Letöltve: 2014. október 30.]
- [9] **Buro, M. - Furtak T., M (2004)** *RTS Games and Real-Time AI Research*. Department of Computing Science, University of Alberta, Edmonton, Canada  
<http://webdocs.cs.ualberta.ca/~furtak/pub/BRIMS-04.pdf>  
[Letöltve: 2014. november 04.]
- [10] **Dan Klein (2010)**: *Decision Diagrams*. Introduction to Artificial Intelligence kurzus, UC

Berkeley, Berkeley, California, USA

<http://inst.eecs.berkeley.edu/~cs188/fa10/slides/FA10%20cs188%20lecture%2018%20--%20decision%20diagrams%20%286PP%29.pdf>

[Letöltve: 2014. október 5.]

**[11] MacQueen, J. B. (1967)** Some Methods for classification and Analysis of Multivariate Observations.

[http://projecteuclid.org/download/pdf\\_1/euclid.bsmmsp/1200512992](http://projecteuclid.org/download/pdf_1/euclid.bsmmsp/1200512992)

[Letöltve: 2014. szeptember 29.]

**[12] Matteucci, M.** A Tutorial on Clustering Algorithms. Politecnico Di Milano  
[http://home.deib.polimi.it/matteucc/Clustering/tutorial\\_html/](http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/)

[Letöltve: 2014. október 10.]

**[13] Unity Technologies.** *Unity dokumentáció.*

<http://docs.unity3d.com/Manual/>

[Letöltve: 2014. szeptember 10.]

**[14] Ortega, J. P. - Rojas, D. R. B. - Garcia, M. J. S.** Research issues on K-means Algorithm: An Experimental Trial Using Matlab.

<http://ceur-ws.org/Vol-534/Paper10.pdf>

[Letöltve: 2014. október 27.]

**[15] Pham D. T. - Dimov S. S. - Nguyen C. D.** Selection of K in K-means Clustering. Manufacturing Engineering Center. Cardiff University, Cardiff, UK  
<http://www.ee.columbia.edu/~dpwe/papers/PhamDN05-kmeans.pdf>

[Letöltve: 2014. november 4.]

**[16] Shachter R. D. - Bhattacharjya D.** Solving influence diagrams: Exact algorithms (2010)

<http://www.it.uu.se/edu/course/homepage/aism/st11/Shachter10.pdf>

[Letöltve: 2014. október 3.]

**[17] Williamson, D. M – Russell G. A – Mislevy R. J.** Model Criticism of Bayesian Networks with Latent Variables (2013)

<http://arxiv.org/ftp/arxiv/papers/1301/1301.3902.pdf>

[Letöltve: 2014. november 8.]