A képen szöveg látható

Automatikusan generált leírás

Malom Játék

Fejlesztői Dokumentáció

Hoki Attila, F2XVKV

Koltai Armand, GYBASN

Kovács Márk, B2TN3S

Orsós János, CEKWOT

Mérnökinformatikus BSc.

Ötödik félév

Mesterséges Intelligencia Alapjai, N-LA03

Haladó Programozás, N-LA05

Szoftvertechnológia, N-LA02

2024

Tartalomjegyzék

[Tartalomjegyzék 1](#_Toc182068097)

[Bevezetés 3](#_Toc182068098)

[1. Követelmények 4](#_Toc182068099)

[1.1. Szoftvertechnológia tárgy követelményei 4](#_Toc182068100)

[1.2. Mesterséges intelligencia tárgy követelményei 4](#_Toc182068101)

[1.3. Haladó programozás tárgy követelményei 4](#_Toc182068102)

[1.4. Követelményanalízis 5](#_Toc182068103)

[2. Koncepcióterv 7](#_Toc182068104)

[2.1. Program képességei 7](#_Toc182068105)

[2.2. Játékmotor szükségessége 8](#_Toc182068106)

[2.3. Grafikus felhasználói felület koncepció 8](#_Toc182068107)

[2.4. Adattárolás és kezelés 9](#_Toc182068108)

[2.5. Kereső algoritmusok 9](#_Toc182068109)

[2.5.1. GreedyPlayer játékosprofil 9](#_Toc182068110)

[2.5.2. A\* játékosprofil 10](#_Toc182068111)

[2.6. Gépi tanulás 11](#_Toc182068112)

[2.7. Tesztelés 11](#_Toc182068113)

[2.8. Hipotézisek 11](#_Toc182068114)

[2.8.1. A gépi tanulást alkalmazó gépi játékos legalább olyan mértékben nyer, mint amelyik játékos az A\* keresést használja. 11](#_Toc182068115)

[2.9. Kiértékelés és eredmény analízis 11](#_Toc182068116)

[3. Rendszerterv 12](#_Toc182068117)

[3.1. Használati eset-diagram (Use-case UML) 12](#_Toc182068118)

[3.1.1. Használati esetek részletes szöveges ismertetése 12](#_Toc182068119)

[3.1.2. Aktorok részletes leírása 12](#_Toc182068120)

[3.2. Tevékenység-diagram (Action UML) 12](#_Toc182068121)

[3.3. Állapotgép diagramok (State Machine) 12](#_Toc182068122)

[3.4. Kontextus diagram (Context diagram) 13](#_Toc182068123)

[3.5. Osztály diagram (Class UML) 13](#_Toc182068124)

[3.6. Adatbázis egyedkapcsolat diagram (ER) 13](#_Toc182068125)

[4. Technológia 14](#_Toc182068126)

[4.1. Fejlesztői környezet 14](#_Toc182068127)

[4.1.1. Microsoft Visual Studio Code 14](#_Toc182068128)

[4.1.2. JetBrains PyCharm: 14](#_Toc182068129)

[4.2. Paradigmák és implementációs szabványok 14](#_Toc182068130)

[4.3. Verziókövetés 14](#_Toc182068131)

[4.4. Programnyelvek 14](#_Toc182068132)

[4.4.1. Python 14](#_Toc182068133)

[4.5. Függvénykönyvtárak 14](#_Toc182068134)

[4.6. Futási környezet 14](#_Toc182068135)

[4.7. Egyéb felhasznált eszközök és szoftverek 15](#_Toc182068136)

[4.7.1. GanttProject 15](#_Toc182068137)

[4.7.2. Software Ideas Modeler 15](#_Toc182068138)

[4.7.3. Figma 15](#_Toc182068139)

[5. Mérföldkövek és feladatok 16](#_Toc182068140)

[5.1. Játék motor 16](#_Toc182068141)

[5.1.1. Logika 16](#_Toc182068142)

[5.1.2. Felhasználói interakció 16](#_Toc182068143)

[5.1.3. Párhuzamos mellékfeladatok 16](#_Toc182068144)

[5.2. Prototípus I. 16](#_Toc182068145)

[5.2.1. Hegymászó algoritmus 16](#_Toc182068146)

[5.2.2. Grafikus felhasználói felület v.0.1 16](#_Toc182068147)

[5.2.3. Testing I. 16](#_Toc182068148)

[5.3. Prototípus II. 17](#_Toc182068149)

[5.3.1. Gépi tanulás 17](#_Toc182068150)

[5.3.2. Grafikus felhasználói felület v.1.0 17](#_Toc182068151)

[5.3.3. Egyéb funkciók 17](#_Toc182068152)

[5.3.4. testing II. 17](#_Toc182068153)

[5.4. Publikálás 17](#_Toc182068154)

[6. Projekttervezés 18](#_Toc182068155)

[6.1. Kockázatterv 18](#_Toc182068156)

[6.1.1. Kockázatelemzési bevezető 18](#_Toc182068157)

[6.1.2. Halszálka-diagram 18](#_Toc182068158)

[6.1.3. Pareto-diagram 19](#_Toc182068159)

[6.1.4. SWOT analízis 20](#_Toc182068160)

[6.2. Csapattagok leterheltsége 21](#_Toc182068161)

[6.3. Ütemterv (Gantt-diagram) 21](#_Toc182068162)

[6.4. Projektszervezési modell kiválasztása 23](#_Toc182068163)

[6.5. Tesztelési terv (Unittest) 23](#_Toc182068164)

[6.5.1. Megközelítés 23](#_Toc182068165)

[6.5.2. Hipotézisek 23](#_Toc182068166)

[6.5.3. Teszt I. 23](#_Toc182068167)

[6.5.4. Teszt elemzés I. 23](#_Toc182068168)

[6.5.5. Teszt II. 23](#_Toc182068169)

[6.5.6. Teszt elemzés II. 23](#_Toc182068170)

[6.6. Projekt lezárása 23](#_Toc182068171)

[6.7. Projekt utóélete 23](#_Toc182068172)

[7. Felhasználói dokumentáció 24](#_Toc182068173)

[7.1. Telepítési útmutató 24](#_Toc182068174)

[7.2. Belépés: Felhasználó létrehozása vagy kiválasztása 24](#_Toc182068175)

[7.3. Főablak 24](#_Toc182068176)

[7.4. Navigációspanel funkciói 24](#_Toc182068177)

[7.5. Ellenfél kiválasztása és nehézség beállítása 24](#_Toc182068178)

[7.6. Hogyan játsz 24](#_Toc182068179)

[7.7. Tippek 24](#_Toc182068180)

[Összefoglalás 25](#_Toc182068181)

[Ábrajegyzék 26](#_Toc182068182)

[Melléklet 27](#_Toc182068183)

Bevezetés

A Malom Játék projekt célja, hogy egy grafikus felülettel rendelkező, mesterséges intelligenciát alkalmazó játékot hozzunk létre, amely a hagyományos malomjáték digitális változatát valósítja meg. A csapatunk négy tagból áll: Hoki Attila, Koltai Armand, Kovács Márk és Orsós János. Mérnökinformatikus BSc. tanulmányaink ötödik félévében, a Mesterséges Intelligencia Alapjai, Haladó Programozás és Szoftvertechnológia tantárgyak keretében dolgozunk a projekten. Célunk egy olyan alkalmazás kifejlesztése, amely lehetőséget biztosít számunkra programozói készségeink fejlesztésére, mindamellett, hogy komplex projektekkel járó menedzsmenti-, és integrációs-, implementációs problémákat is megtanuljunk kezelni és elhárítani, de legfőképp mindenekelőtt megelőzni.

A projekt során több kihívással is várhatóan szembe kell majd nézzünk. Elsőként fontos, hogy a grafikus felhasználói felület tervezése és implementálása során a felhasználók számára intuitív és élvezetes játékkörnyezetet biztosítsunk. Emellett a mesterséges intelligencia integrálása is kulcsfontosságú, hiszen az M.I.-nek képesnek kell lennie arra, hogy hatékonyan versenyezzen a játékosok ellen és folyamatosan alkalmazkodjon a játék dinamikájához. A projekt során kiemelt figyelmet fordítunk a kódminőségre és a programozási elvek betartására, hogy a rendszer később is könnyen karbantartható és bővíthető legyen.

A bevezetőben említett megközelítésünk célja nemcsak a játék megalkotása, hanem a három tantárgy követelményeinek teljesítése is. Ennek érdekében a következő lépéseket tervezzük a fejlesztés mellett elvégezni: részletes követelményanalízis, a projekt koncepciójának kidolgozása, a fejlesztési folyamat dokumentálása, és végül a játék bemutatása a 2024/2025-1 tanév szorgalmi időszakának 12. hetében.

A Malom Játék projekt tehát számunkra egy átfogó lehetőséget kínál, hogy a különböző tantárgyi elvárásoknak megfelelve egy innovatív, kihívásokkal teli és tanulságos fejlesztési folyamatot valósítsunk meg. E projekt keretében nemcsak a játékkal kapcsolatos tudásunkat és készségeinket bővíthetjük, hanem a csapatmunka és a problémamegoldás terén is fejlődhetünk.

1. Követelmények

Grafikus felülettel rendelkező, gépi tanulást alkalmazó rendszer fejlesztése, dokumentálása és a szoftver bemutatása a 12.héten a 2024/2025-1 tanév szorgalmi időszakában az alábbi tantárgyak követelményeit szem előtt tartva.

* 1. Szoftvertechnológia tárgy követelményei

Az 1. számú melléklet tartalmazza a követelménylista eredeti példányát.

* Projektterv MS Projektben.
* Projekt kockázatok elemzése (halszálka, kockázat értékelés + Pareto, kockázat tervezés (stratégia), SWOT).
* A szoftver által támogatandó tevékenység/funkcionalitás szabad szöveges leírása.
* Táblázatos rendszerezés (strukturált szöveg).
* Használati eset diagram.
* Használati esetek részletes szöveges ismertetése.
* Aktorok részletes leírása (ld. gyakorlat táblázatai).
* Tevékenység diagramok az egyes használati esetekhez.
* Állapot gép diagramok.
* Kontextus diagram.
* Szakarchitektúra diagram.
  1. Mesterséges intelligencia tárgy követelményei

A 2. számú melléklet tartalmazza a követelménylista eredeti példányát. A feladat egy mesterséges intelligencia alapú alkalmazás elkészítése egy szabadon választott témában, amelyet egy kiválasztott modell segítségével kell megoldani. A megoldáshoz felhasználóifelületet is létre kell hozni, ahol az alkalmazás működése bemutatható.

* Megfelelő téma
* Gépi tanulási vagy Mély tanulási megoldás
* Egyszerűbb felhasználói felület
* GitHub Repository, ahol csapatok minden tagja commit-telt
* Körülbelül 5 oldalas beszámoló munkájukról
  1. Haladó programozás tárgy követelményei

Mesterséges Intelligencia tárgy elvárásaival megegyezik.

* 1. Követelményanalízis

A Szoftvertechnológia tantárgy keretein belül a projekt tervezése és megvalósítása során fontos a részletes dokumentáció és a projektmenedzsment módszerek alkalmazása. A projektterv elkészítése, amely MS Projektben valósul meg, lehetővé teszi a fejlesztési fázisok, feladatok és határidők precíz nyomon követését. A jól kidolgozott projektterv alapvetően meghatározza a fejlesztés menetét, és segít a csapat számára a célok elérésében.

A kockázatelemzés különösen fontos szakasza a projektnek. A halszálka-diagram segítségével a potenciális problémák és azok okainak azonosítása válik lehetővé. Ez a vizuális eszköz segít a csapatnak abban, hogy proaktívan felkészüljön a nehézségekre, és kidolgozza a megfelelő válaszlépéseket. A Pareto-diagram alkalmazása lehetővé teszi a kockázatok priorizálását, így a csapat a legkritikusabb problémákra tud fókuszálni, míg a SWOT-elemzés segítségével a projekt erősségeit és gyengeségeit is felmérhetjük.

A szoftver által támogatandó tevékenységek és funkciók szabad szöveges leírása, valamint táblázatos rendszerezése segíti a projekt áttekinthetőségét és a csapat tagjainak közötti információmegosztást. A használati eset diagramok és a hozzájuk tartozó részletes szöveges ismertetés bemutatja, hogyan interakcióba lépnek a felhasználók a rendszerrel, így megkönnyítve a tervezési fázist. Az aktorok részletes leírása és a tevékenységdiagramok az egyes funkciók működését modellezik, ezáltal biztosítva a projekt teljes körű megértését. Az állapotgép diagramok pedig a rendszer működésének dinamikáját ábrázolják, lehetővé téve a lehetséges hibák és problémák előrejelzését.

A Mesterséges Intelligencia tantárgy követelményei külön hangsúlyt fektetnek a gépi tanulási megoldások alkalmazására. A projekt keretében egy mesterséges intelligencia alapú alkalmazást kell készíteni, amely egy szabadon választott témát céloz meg. Ez a követelmény lehetőséget ad a csapatnak, hogy innovatív módon közelítse meg a választott problémát, valamint hogy olyan megoldást találjon, amely a gépi tanulási modellek előnyeit maximálisan kihasználja.

A téma megfelelő kiválasztása kulcsfontosságú, mivel a projekt sikeressége szorosan összefügg azzal, hogy mennyire releváns és aktuális a megcélzott probléma. A gépi tanulási vagy mély tanulási megoldás alkalmazása elengedhetetlen, mivel ezek a technológiák a modern szoftverfejlesztés alapkövei. A felhasználói felület létrehozása szintén kiemelt jelentőségű, hiszen az alkalmazás működésének bemutatása révén a felhasználói élmény és a hatékonyság nagymértékben függ a felület minőségétől és funkcionalitásától.

A GitHub repository használata nemcsak a csapatmunka hatékonyságát növeli, hanem segíti a projekt nyomon követését és a verziókezelést is. A rendszeres commitok révén a csapat tagjai folyamatosan frissíthetik a munkájukat, ami elősegíti a gördülékeny együttműködést. A körülbelül 5 oldalas beszámoló elkészítése pedig lehetőséget ad arra, hogy a csapat részletesen bemutassa az alkalmazás működését, a választott AI modellek eredményeit és a fejlesztési folyamat során szerzett tapasztalatokat

A Haladó Programozás tantárgy elvárásai nagymértékben hasonlóak a Mesterséges Intelligencia tantárgyéval. Itt is elengedhetetlen a megfelelő gépi tanulási modell alkalmazása, valamint a felhasználói felület fejlesztése. A hangsúly a programozás minőségén és a technikai megoldások innovativitásán van. A csapatnak figyelmet kell fordítania a kód olvashatóságára, a hatékonyságra és a moduláris felépítésre, hogy a rendszer a későbbiekben is könnyen karbantartható legyen.

A projekt során a csapatnak szorosan együtt kell működnie, hogy minden tantárgyi követelménynek eleget tegyen. Az eltérő tantárgyak követelményei közötti összehangolás nemcsak a projekt sikeressége szempontjából kulcsfontosságú, hanem hozzájárul a csapat szakmai fejlődéséhez is. A kihívások és nehézségek kezelése során a csapat tagjai új készségeket sajátíthatnak el, amelyek a jövőbeni projektjeik során is hasznosak lesznek.

1. Koncepcióterv

A koncepcióról részletesebben az 5. mellékletben megadott dokumentumban olvasható.

* 1. Program képességei

Célunk egy olyan malomjáték grafikus asztali alkalmazás fejlesztése, amely első sorban a gép tanulási készségeinket hivatott fejleszteni, azonban mint önálló játék is megállja a helyét személyre szabhatóságának és széles skálájú funkcionalitásainak köszönhetően.

Alapvetően a tantárgyak meghatározott követelmények kielégítésén felül szeretnénk, hogy az alkalmazás minél nagyobb felhasználói élményt nyújtson, így egy intuitívan használható, és a játékos ügyességi szintjéhez igazítható programot szeretnénk létrehozni. Ezt úgy érhetjük el, hogy egy egyszerű, letisztult, minimalista és átlátható felületet biztosítunk, ahol a keresett funkciók intuitívan fellelhetőek és beállíthatóak.

Ilyen beállítási lehetőség lenne, például a játékmód, amelyet szeretnék három koncepcióban is működőképessé tenni, mégpedig az személy-személy ellen, személy-gép ellen, és végül gép-gép ellen. Ezek hatékony bemutatására kellően rövid, ám kimerítő demóval készülünk.

A játékmódválasztáson felül fontosnak éreztük a gépi játékosok esetében is a személyre szabhatóságot előtérbe helyezni, így az is megválasztható, hogy milyen nehézségi szinten szeretnénk játszani, amely végeredményében sokszor például a mélységi korlát indirekt manipulációját fogja jelenteni a gráfbejáró algoritmusok esetében.

Szeretnénk lehetővé tenné a gépi tanulási modellünk folyamatos továbbfejlesztését, amelyet úgy tehetünk lehetővé, hogy még több adatot biztosítunk a tanuláshoz. Minden egyes játék megfelelően archiválható és az újra-tanításhoz felhasználható. Emellett fontosnak tartjuk a modellvalidációt és a játékosok tudtára adni teljesítményüket, így egy statisztikai menüpontot is kívánunk megvalósítani, ahol a többi játékossal összehasonlíthatjuk mind a saját-, és a gépi modell teljesítményét is grafikus módszerekkel.

Az alkalmazáson belüli statisztikára exportálási lehetőséget is szeretnénk biztosítani, elsősorban a dokumentáció megkönnyítéséhez.

Összességében egy kellően komplex, funkciókkal telített, kihívásokkal teli projektet szeretnénk megvalósítani, amit mindig büszkén fel tudunk majd mutatni képességeink igazolására.

* 1. Játékmotor szükségessége

Az ötletelési fázisban, a specifikációk meghatározásakor arra a döntésre jutottunk, hogy mielőtt belevetnénk magunkat a gépi tanulás világába, célszerű lenne egy stabilan működő, játszható játékot készíteni, hogy ezzel is a hibaforrások számát redukáljuk, és megkönnyítsük a dolgunkat a későbbiekben a fejlesztés folyamán.

Alapvetően az első mérföldkő célja, hogy a játékmotor funkcionálisan üzemeljen olyan szinten, hogy magabiztosan lehessen hivatkozni rá és integrálni mind a grafikus interfész, mind a gépi ellenfél fejlesztése során.

A motor önmagában is játszható konzol-interfészen keresztül, emellett „debug” módban is futtattható, hogy később a szimulációkat indíthassuk és statisztikát készíthessünk.

* 1. Grafikus felhasználói felület koncepció

Az alap koncepció egy könnyen átlátható és navigálható ablak létrehozása, amely egy egyszerű menüszalaggal rendelkezik. Az ablak központjában természetesen a játéktábla kell, hogy álljon, azonban érdekesnek gondoltuk az eddigi lépések vizualizálást is egy oldalpanelen. Ez kifejezetten akkor érdekes, amikor a játékos a gép ellen játszik, és a morálra érdekesen hathat, amikor a játék felénél azon kapja magát, hogy a gép jóslata a lépéseinket illetően minden esetben talált.

Nem maradhat el a kötelező kezdő lépési jog sorsolása sem, így a belépéskor „dobunk” is egy képzeletbeli kockával. Hogy megismételhető legyen egy-egy játszma, így a „seed”-et a random generátorok esetében szintén mellékeljük az aktuális játék mentésében.

Az ötletelés során megszületett koncepciót tovább csiszolva született meg az elsőiterációs látványterv a Figma webes eszköz segítségével. A grafikai elemek változtatásának jogát a fejlesztés ezen fázisában fenntartjuk. (1.ábra)

A screenshot of a computer

Description automatically generated

ábra : Megjelenítő felület kezdeti és kiforrottabb koncepciós vázlata

* 1. Adattárolás és kezelés

Az adatok tárolásának formátuma és struktúrája gondosan megtervezett, hogy lehetővé tegye a játékok rögzítését, elemzését és újra-tanításhoz való felhasználását. A struktúra biztosítja a gyors elérést és a hatékony feldolgozást.

Az adatokat log fájlokban tároljuk, melyek tartalmazzák a játék egyedi azonosítóját, és hogy milyen eredménnyel zárult a játék (győzelem, döntetlen, feladott, megszakított). Mivel ezeket az adatokat tanításra is felhasználjuk, ezért beépített takarító eljárással megtisztítjuk az adatokat minden játék kezdetén, ahol a be nem fejezett játékokat töröljük az adatbázisból.

* 1. Kereső algoritmusok

A játék fejlesztésében hegymászó és A\* keresési algoritmusokat alkalmazunk. Ezek használatát az indokolja, hogy az algoritmusok egyre nagyobb komplexitást kínálnak, így a projekt folyamatos fejlesztése során is bővíthetők.

Az A\* mivel legoptimálisabb játékos, ezért az általa játszott mohó algoritmust megvalósító gépi játékos ellen játszott játszmákkal fog megtörténni a gépi tanulási modellünk feltanítása.

* + 1. GreedyPlayer játékosprofil

A GreedyPlayer játékosprofil a mesterséges intelligencia tárgyköréből is ismert mohó megközelítést alkalmazza, vagyis lokális optimum alapján dönt egy-egy lépés mellett vagy ellen. Az algoritmus nem tartja számon a korábbi lépéseket, valamint nem „tekint előre”, azaz nem számol az egyes lépések későbbi hatásaival. Ha szigorúan a mesterséges intelligencia terminológiájával szeretnénk definiálni, akkor a modell olyan állapotfával dolgozik melynek mélysége 1.

Habár a Malom játékmenete lerakási és mozgatási fázisból is áll a lokális optimum definiálásánál ez nem feltétlen okoz különbséget, egyedül a lépésért felelős függvény visszatérési értéke fog változni (egy mezőkoordináta a lerakási fázisban, két koordináta a mozgatási fázisban [honnan ↔ hova]). Előnyben részesíti azokat a lépéseket melyekkel malomba lépünk, vagy ellenséges malmot tudunk gátolni, azonban e között a két lépéstípus között is különbséget tesz. A malom generáló lépések prioritást élveznek, hiszen zérus összegű játék révén minden általunk szerzett előny az ellenfél hátrányként ütközik ki, ennek megfelelően a modell az előbbi típusok vizsgálatát folytatja első körben, ha nem talál ilyet akkor kezdődik meg a malmot gátló lépések kiértékelése. Ha sem az előbbi sem az utóbbi típusból nem áll rendelkezésre egy lépés sem, akkor a modell véletlenszerűen választ egy A képen szöveg, képernyőkép, diagram, Betűtípus látható

Automatikusan generált leírásopciót a játékmotor által biztosított szabályos lépések közül.

ábra 2: GreedyPlayer algoritmikus felépítése (Flow Chart)

* + 1. A\* játékosprofil

A screenshot of a computer

Description automatically generatedAlapvetően az A\* algoritmus a mohó továbbfejlesztése. A hatékonyság növelése itt abban mutatkozik meg, hogy heurisztikát szolgáltató döntési fával egészül ki az algoritmus, ami Kovács Márk játékstílusát hivatott szimulálni. A döntési fának a működését a (2.ábra) mutatja.

ábra 3: A\* algoritmus heurisztikájául szolgáló döntési fa

Nyolc döntési feltételből áll, amelyek a védekezést helyezik előtérbe. Minél lentebb haladunk a döntési fán, azaz nincs ok a védekezésre, akkor offenzív módba vált. A támadási metódusok a legjövedelmezőbbtől a közel véletlenszerűsített lépésig megy.

A lehelyezési és a mozgási fázisok között az eljárás nem tesz különbséget. Gyorsítani lehetne a keresést, a külön eljárásokkal bővítenénk a játék fázisainak függvényében, mozgási heurisztikákkal, de jelen állapotában is kellően hatékony.

* 1. Gépi tanulás

A gépi tanulás alapelveként az egyszerűség és gyors implementáció érdekében a modell egyszerűbb algoritmusokra épül, nem deep learning alapú. A tanítási terv egy iteratív megközelítést követ, amely lehetővé teszi az algoritmus folyamatos fejlesztését.

To be continued…

* 1. Tesztelés

A projekt tesztelési folyamatait a különböző fejlesztési szakaszokban végezzük. A tesztelés a funkcionális és teljesítménybeli követelmények igazolását szolgálja. Részletes tesztelési tervet a későbbi szekciókban olvashat (6.5).

* 1. Hipotézisek
     1. A gépi tanulást alkalmazó gépi játékos legalább olyan mértékben nyer, mint amelyik játékos az A\* keresést használja.

A gépi tanulás alkalmazása során a modell folyamatosan fejlődik és tanul az előző játékokból, ezáltal képes lehet olyan lépéseket azonosítani, amelyek növelik a nyerési esélyeit, és hosszú távon a döntései hatékonyabbá válnak.

A hipotézis igazolására több játszmát futtatunk mindkét algoritmussal, majd statisztikai elemzést végzünk a nyerési arányok összehasonlítására. A mérések során figyelmet fordítunk az ellenfél típusa és a játszott nehézségi szint hatásaira is.

* 1. Kiértékelés és eredmény analízis

Az elemzések exportálására külön funkciókat biztosítunk, amelyek támogatják a szimulációs környezetben létrejövő adatok újra hasznosítását és az eredmények vizualizációját, így gyors és valós adatokkal dolgozhatunk.

1. Rendszerterv
   1. Használati eset-diagram (Use-case UML)
      1. Használati esetek részletes szöveges ismertetése
      2. Aktorok részletes leírása
   2. Tevékenység-diagram (Action UML)
   3. Állapotgép diagramok (State Machine)

Az alábbi ábrán a játék teljes állapotgép diagramja látható a felhasználó szempontjából, amely végig kíséri a játék letöltésétől az egyik fél győzelmének vagy döntetlen helyzet kialakulásának állapotáig. A kritikus szekciók a letöltés, módválasztás és a második játékbéli fázis.

A diagram of a company

Description automatically generated with medium confidence

ábra : A játék teljes felhasználói állapotgép diagramja

* 1. Kontextus diagram (Context diagram)

A white square with black text

Description automatically generated

ábra : A malomjáték szoftver kontextus diagramja

3.1. Táblázat: Fejlesztő aktor definiálása

|  |  |
| --- | --- |
| **Fejlesztő (tesztelő)** | |
| Feladat | Konzolon keresztül szimulációk futtatása, modell tanítás, adatgenerálás. |
| Mennyiség | Kéthetente egy alkalom. |
| Fajta | Alkalmazott |
| Betanítási idő | 1. hét |

3.2. Táblázat: Játékos aktor definiálása

|  |  |
| --- | --- |
| **Játékos (ember)** | |
| Feladat | Program futtatása grafikus interfészen. Játlék PVP-, vagy PVC módokban. |
| Mennyiség | \* |
| Fajta | Természetes személy |
| Betanítási idő | - |

* 1. Osztály diagram (Class UML)
  2. Adatbázis egyedkapcsolat diagram (ER)

1. Technológia
   1. Fejlesztői környezet
      1. Microsoft Visual Studio Code

* Verzió: 1.94.2
* Build: 10262041
* 2024.09
  + 1. JetBrains PyCharm:
* Verzió: 2024.2.3
* Build: 242.23339.19
* 2024.09.26
  1. Paradigmák és implementációs szabványok

Objektum orientált programozás, és modularitás, portolhatóság priorizálása.

* 1. Verziókövetés
* Git Verzió: 2.47.0
* GitHub Repository: https://github.com/hokiattila/Malom\_Game\_NJE.git
  1. Programnyelvek
     1. Python
* Verzió: 3.13.0
* 2024.10.07
  1. Függvénykönyvtárak
* matplotlib,
* pandas,
* numpy,
* tkinter,
* customtkinter,
* unittest,
* pillow
  1. Futási környezet
* Windows 10 Pro 22H2
* Windows 11 Pro 23H2
  1. Egyéb felhasznált eszközök és szoftverek
     1. GanttProject
* Verzió: 3.3.3312
* 2024.01.15
  + 1. Software Ideas Modeler
* Verzió: 14.70
* 2024.10.08
  + 1. Figma
* Vezrió: 124.4.7
* 2024.09.18.

1. Mérföldkövek és feladatok
   1. Játék motor

A játék alapjául szolgáló motor fejlesztése, amely magában foglalja a játékmenet logikáját, a felhasználói interakciók kezelését és a fejlesztéshez kapcsolódó kiegészítő feladatokat.

* + 1. Logika

A játék működését meghatározó alapvető logikai szabályok kidolgozása és az ehhez kapcsolódó osztályok, metódusok létrehozása. CLI (Command Line Interface) megjelenítés implementálása a játék egyszerűsített teszteléséhez.

* + 1. Felhasználói interakció

Alapvető PvP (Player vs Player) funkció kialakítása, hogy a játék két játékos által játszható legyen. Egy későbbi szakaszban egy debug mód beépítése, amely segíti a fejlesztők munkáját és a hibakeresést.

* + 1. Párhuzamos mellékfeladatok

Dokumentáció indítása, amely folyamatosan rögzíti a fejlesztés lépéseit, döntéseit. GUI (grafikus felhasználói felület) elsődleges makettjének (mockup) készítése, mely a felhasználói élmény és elrendezés alapját képezi.

* 1. Prototípus I.

A játék első prototípusának megvalósítása, amely már tartalmazza a kezdeti algoritmusokat és a grafikus felületet.

* + 1. Hegymászó algoritmus

A heurisztikus keresési algoritmus megvalósítása, amely backtrackinggel és anélkül is működhet. Mélységi korlátok bevezetése, amelyek szabályozzák az algoritmus keresési mélységét és hatékonyságát.

* + 1. Grafikus felhasználói felület v.0.1

A GUI mockup alapján elkészített felület első iterációja.

* + 1. Testing I.

A játék motor kezdeti tesztelése, amely megerősíti a játékmenet működését és stabilitását. Részletesebb tesztelési terv kidolgozása, amelyet a 6.5.3. pont tartalmaz.

* 1. Prototípus II.

A második prototípus, amelyben a mesterséges intelligencia algoritmusai, a GUI továbbfejlesztése, valamint kiegészítő funkciók jelennek meg.

* + 1. Gépi tanulás

A\* algoritmust alkalmazó játékos megvalósítása, és felhasználása betanítási adat (training data) generálására. Gépi tanulást alkalmazó játékos fejlesztésének lépéseinek kidolgozása későbbi implementáláshoz.

* + 1. Grafikus felhasználói felület v.1.0

Teljes funkcionalitású, letisztult GUI, amely zökkenőmentesen kezeli az adatáramlást a játék motor, a GUI és az adatbázis között. A felhasználói dokumentációban részletesen kifejtett interfész és működés.

* + 1. Egyéb funkciók

Felhasználói kezelés (user handling), amely lehetővé teszi a felhasználók kezelését és nyomon követését. Jelentés- és exportálási funkciók létrehozása, amelyek lehetővé teszik az adatok mentését és elemzését. Statisztikák generálása a játékmenet elemzéséhez és értékeléséhez.

* + 1. testing II.

Sebességtesztek elvégzése, amelyek biztosítják a játék megfelelő válaszidejét és teljesítményét. GUI v1.0 tesztelése, amely magában foglalja a felhasználói élmény vizsgálatát, részletes tesztelési terv alapján, melyet az 5.3.3 pont tartalmaz

* 1. Publikálás

Prezihez demo, installer,

1. Projekttervezés
   1. Kockázatterv
      1. Kockázatelemzési bevezető

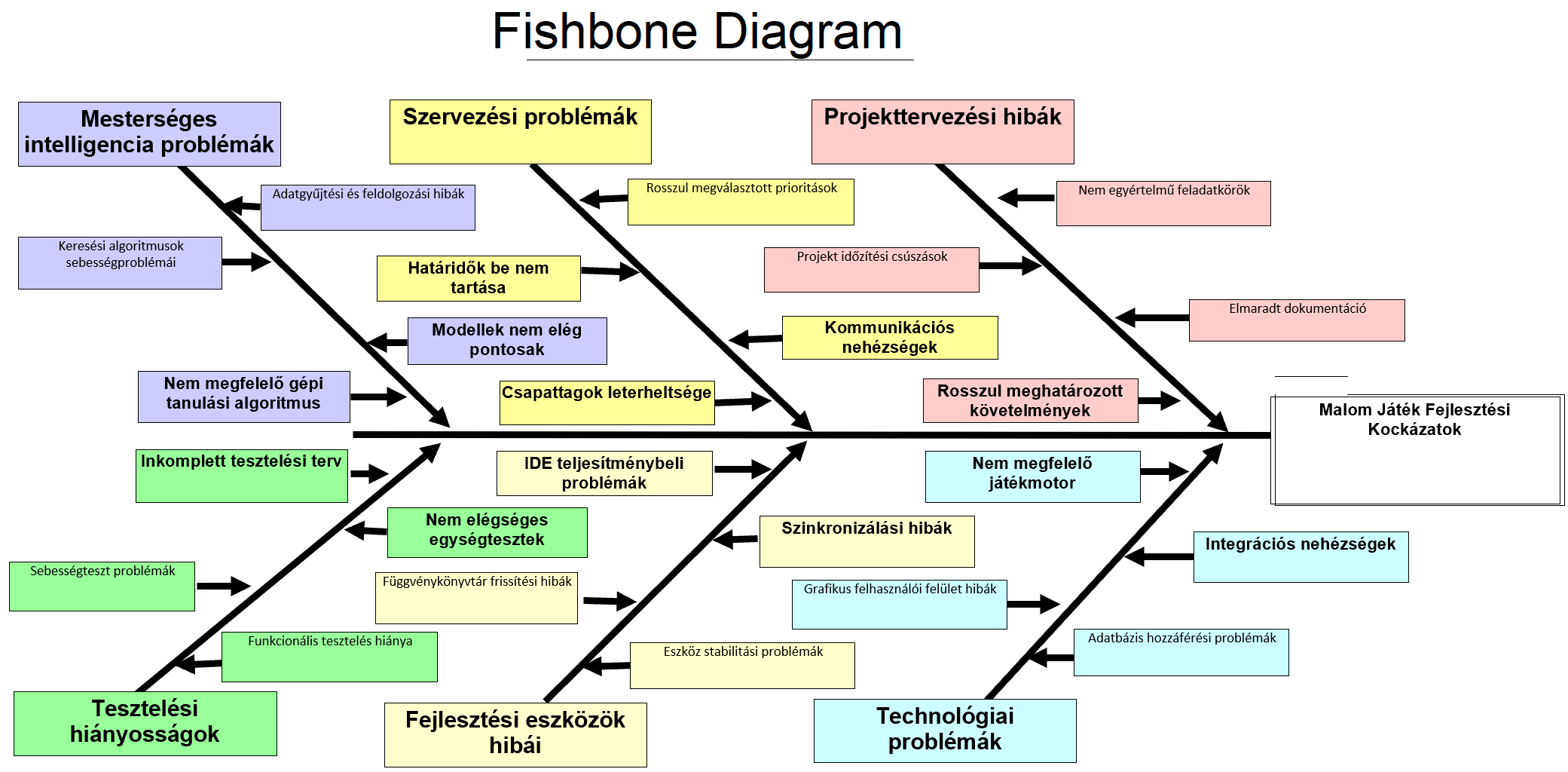
A három különböző tantárgyra készített projekt jelentős kockázatot hordoz, mivel a követelmények összehangolása, a különböző elvárások kielégítése és a megnövekedett időigény jelentős tervezési és fejlesztési többletet eredményez. A siker kulcsa a részletes előzetes tervezés, a követelmények pontos megértése és az, hogy a projekt minden szempontból megfeleljen a különböző tantárgyi elvárásoknak.

A kockázati tervezés során különféle bevált módszereket, például halszálka-diagramot, Pareto-diagramot és SWOT elemzést alkalmazunk annak érdekében, hogy feltárjuk azokat a potenciális kudarctényezőket, amelyeket az ötletelés során esetleg figyelmen kívül hagytunk. Egy alapos elemzéssel biztosíthatjuk, hogy a fejlesztés kritikus feladatai során körültekintően járjunk el, és folyamatosan figyelembe vegyük a projekt sikerét veszélyeztető lehetséges kockázatokat.

* + 1. Halszálka-diagram

A projekttervezési hibák a rosszul meghatározott specifikációkból, az időzítési csúszásokból és a nem megfelelő kommunikációból erednek, amelyek gátolhatják a Malom játék fejlesztésének sikerét. A technológiai problémák, például a játékmotor stabilitása, az Interfész hibák, valamint az adatbázis kezelés kihívásai szintén nagyban akadályozhatják a projekt előrehaladását. A szervezési nehézségek, mint a csapattagok leterheltsége és a kommunikációs akadályok, további kockázatokat jelentenek. A fejlesztési eszközök hibái, például az IDE stabilitási problémák vagy a verziókövetési rendszerek megfelelőtlen használata jelentős technikai nehézségeket okozhatnak a csapatnak. A mesterséges intelligenciával kapcsolatos kihívások, mint a nem megfelelő algoritmusok, a lassú keresési eljárások és a modellek pontatlansága további fejlesztési akadályokat jelenthetnek. Végül, a nem megfelelő tesztelés, különösen a hiányos egység- és sebességtesztek, rejtett hibákhoz vezethetnek.

Az alábbi halszálka-diagram átfogó képet ad a projekt során felmerülő kockázatokról, és segíthet a csapatnak abban, hogy hatékonyan kezelje ezeket a problémákat a projekt előrehaladása során.



ábra : Malom Játék Fejlesztési Kockázatainak Halszálka-diagramja

* + 1. Pareto-diagram

Az alábbi diagram remekül megmutatja, hogy mik azok a kockázatok, amelyekre különösen oda kell figyelnünk, különben súlyos következménnyel járhat. A kockázatok valószínűségét és hatását ötös értékelési skálán mérve, azok szorzataként megkapjuk az egyes veszélyek aritmetikai súlyosságát. Az így keletkezett viszonyszám segítséget ad a korábban definiált kockázatok pontosabb kategorizálásában, hogy végeredményében a lehető leghatékonyabban előzzük meg. Legjelentősebb célkitűzésünknek a határidők pontos betartásának kell lennie, ugyanis ennek elmarasztása közel katasztrofális hatást jelentene a projekt számára. Fontos megjegyezni, hogy egyben ez a legvalószínűbb és legjelentősebb, így különös figyelmet kell szentelnünk egyén szinten az időmenedzsmentre. Az egyes kockázatonkénti súlyozások és egyéb információk megtalálhatóak a hatodik mellékletben.

ábra : Pareto-diagram a főbb kockázatokról

* + 1. SWOT analízis

A projekt erősségei közé tartozik a jól összeszokott csapat, az egyértelmű célok és a problémamegoldó készség, amelyeket egy ambiciózus projektötlet támogat. Ugyanakkor gyengeségként jelentkezik a tapasztalatlanság a Python nyelvvel, a gépi tanulás terén, valamint a grafikai készségek hiánya. A projekt lehetőséget ad arra, hogy három tantárgy követelményeit egy projektben teljesítsük, egy innovatív koncepcióval, amely továbbfejlesztési potenciált is hordoz magában. A legnagyobb veszélyek a határidőcsúszások, technológiai integrációs komplikációk és a szervezetlenség okozta problémák. A táblázat megtalálható a hetedik melléketben.

A group of colorful rectangular boxes with white text

Description automatically generated

ábra : A FreeKredit fejlesztőcsapat SWOT elemzése

* 1. Csapattagok leterheltsége

Szerencsénkre a projekt komplexitásával arányosan sok allokálható erőforrás áll rendelkezésre, így a feladatok megosztását inkább az egyes fejlesztők készségei és kívánságaikat is szem előtt tartva osztottuk meg, ügyelve a közel egyenlő terhelésre. Az agilis szervezésnek köszönhetően mindenki akkor végzi el a feladatait, amikor időt tud rászánni. Ez a nagyfokú szabadság előnyökkel, de természetesen sok hátránnyal is jár, ugyanis minden résztvevőnek fejlett időmenedzsment készségekre, de legfőképpen önfegyelemre lesz szüksége az eredményes munkavégzéshez. Az alábbi erőforrásigénybevételi ábrázolás jól mutatja, hogy a feladatmegosztás sikeres volt, és még a projektvezetőnek sem túlterheltséggel küzdenie. (5. ábra)

A screenshot of a computer

Description automatically generated

ábra : Fejlesztői csapat erőforrás-diagramja

* 1. Ütemterv (Gantt-diagram)

Kardinális szerepet játszik projektünk sikerességében a játék motorjának megalkotása, ugyanis az alapvető logikák, szabályrendszerek megléte nélkül nem tudunk rá hatékony gépi tanulási modellt alkotni, amely a követelmények között az egyik legfontosabb. A motor fejlesztésével párhuzamosan zajlik a dokumentációs struktúra kialakítása, specifikáció meghatározása, követelményanalízis, és az első mérföldkőig elvégzett teendők összefoglalása. (6. ábra)

A screenshot of a graph

Description automatically generated

ábra : Gantt-diagram az első mérföldkőig

A második ütemben még a mesterséges intelligenciához vezető kezdeti algoritmusok implementálására helyezzük a hangsúlyt, mint a „hegymászó” módszer és népszerűbb gráfbejárási eljárások. Ezzel párhuzamosan a grafikus interfész fejlesztése is kezdetét veszi a koncepciós tervek alapján. Kiemeleten fontos a motor megbízható működése, így arra irányulóan elengedhetetlen annak kimerítő tesztelése. Előkészítésképpen a gépi tanuláshoz a korábban említett algoritmusok által szimulált játszmák adatát kell szervezett formában mentenünk. (7. ábra)

A screenshot of a computer

Description automatically generated

ábra : Gantt-diagram a második mérföldkőig

A harmadik fázisban a középpontban a gépi tanulás implementálása lesz a szoftverünkbe, amely egyben az egyik legnagyobb kihívást is jelenti. Természetesen ekkor nem maradhat el a grafikus interfész tesztelése, és az ahhoz kapcsolódó hibák mielőbbi javítása sem. Nélkülözhetetlen információt jelent a gépi tanulási modellünk hatékonyságának mérése, így különböző adatelemzési, statisztikai módszerekkel automatizált jelentés-generálást kell integrálnunk, hogy az egyes tanítási esetek után milyen változások tapasztalhatóak. A harmadik mérföldkő végére a projektünk fő célját elértük, és a továbbiakban a projekt lezárásának előkészítésére koncentráljuk figyelmünket. (8.ábra)

A screenshot of a computer

Description automatically generated

ábra : Gantt-diagram a harmadik mérföldkőig

Az utolsó fázisban, a projekt lezárására törekszünk, ami magába foglalja a program kompresszióját, optimalizációját és portolhatóságát. A munkánk minél hatásosabb demonstrálása érdekében egy játék demóval készülünk, amelyet egy ízléses, lényegre törő prezentációval támogatunk meg. Ezzel páhuzamosan a felhasználói dokumentációt is elkészítjük. A negyedik sprint végén publikáljuk a szoftver első stabil verzióját. (9.ábra)

A screenshot of a computer

Description automatically generated

ábra : Gantt-diagram a negyedik mérföldkőig

* 1. Projektszervezési modell kiválasztása

Agilis modell, azon belül scrum. Gyakran szállítunk (kéthetente new stable build), személyes meetingek (standupok) két ehtente az egész csapatnak, alcsapatokon belül tetszés szerint, mérce a működő softver, a követelmények változhatnak, törekvés az egyszerűségre, letisztultságra, átláthatóságra, rücksprache rundék a projektleaddel, párhuzamosított tevékenységek, inkrementális fejlesztés (engine, gui, ai, full game),

* 1. Tesztelési terv (Unittest)
     1. Megközelítés
     2. Hipotézisek
     3. Teszt I.
     4. Teszt elemzés I.
     5. Teszt II.
     6. Teszt elemzés II.
  2. Projekt lezárása

Lezáró értekezlet, belsős projekt értékelés és visszacsatolás,

* 1. Projekt utóélete

open source github repo, karbantartás, Steam feltöltés (free to play)

1. Felhasználói dokumentáció

Kell. Méghozzá 5 oldal.

* 1. Telepítési útmutató

(ha lesz ilyen)

* 1. Belépés: Felhasználó létrehozása vagy kiválasztása

Login vagy create new user

* 1. Főablak

Játéktér, real time játék history, aktuális játék adatok, „roll” hogy ki kezdjen, navigációs panel

* 1. Navigációspanel funkciói

Ai mode (módok, nehézségek),statistics (with scoreboard), game history,logout

* 1. Ellenfél kiválasztása és nehézség beállítása

Hegymászó, szélességi/mélységi/A\*, ML

Mélységi korlát => nehézség

* 1. Hogyan játsz

New game workflow

* 1. Tippek

Minél többet játszol, annál többet fogsz veszíteni

Összefoglalás

Kihívások, eredmények, fejlesztési lehetőségek

Ábrajegyzék

[ábra 1: Megjelenítő felület kezdeti és kiforrottabb koncepciós vázlata 8](#_Toc181784274)

[ábra 2: A játék teljes felhasználói állapotgép diagramja 11](#_Toc181784275)

[ábra 3: A malomjáték szoftver kontextus diagramja 12](#_Toc181784276)

[ábra 4: Malom Játék Fejlesztési Kockázatainak Halszálka-diagramja 18](#_Toc181784277)

[ábra 5: Pareto-diagram a főbb kockázatokról 19](#_Toc181784278)

[ábra 6: A FreeKredit fejlesztőcsapat SWOT elemzése 19](#_Toc181784279)

[ábra 7: Fejlesztői csapat erőforrás-diagramja 20](#_Toc181784280)

[ábra 8: Gantt-diagram az első mérföldkőig 20](#_Toc181784281)

[ábra 9: Gantt-diagram a második mérföldkőig 21](#_Toc181784282)

[ábra 10: Gantt-diagram a harmadik mérföldkőig 21](#_Toc181784283)

[ábra 11: Gantt-diagram a negyedik mérföldkőig 21](#_Toc181784284)

Melléklet

1. 01\_Projektfeladat\_pontozasa\_Nappali\_Szoftvertechnologia.docx
2. 02\_BEADANDO\_LEIRAS\_MESTERSEGES\_INTELLIGENCIA\_ALAPJAI.pdf
3. 03\_Halszalka\_Sablon\_1.xls
4. 04\_Malom\_Game\_NJE\_GanttProject.gan
5. 05\_Project\_whiteboarding.pdf
6. 06\_Pareto\_diagram.xlsx
7. 07\_SWOT\_analizis.xlsx
8. 08\_Malom\_Project.simp