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](#_Toc183455434)

[Bevezetés 3](#_Toc183455435)

[1. Követelmények 4](#_Toc183455436)

[1.1. Szoftvertechnológia tárgy követelményei 4](#_Toc183455437)

[1.2. Mesterséges intelligencia tárgy követelményei 4](#_Toc183455438)

[1.3. Haladó programozás tárgy követelményei 4](#_Toc183455439)

[1.4. Követelményanalízis 5](#_Toc183455440)

[2. Koncepcióterv 7](#_Toc183455441)

[2.1. Program képességei 7](#_Toc183455442)

[2.2. Játékmotor szükségessége 8](#_Toc183455443)

[2.3. Grafikus felhasználói felület koncepció 8](#_Toc183455444)

[2.4. Adattárolás és kezelés 9](#_Toc183455445)

[2.5. Kereső algoritmusok 9](#_Toc183455446)

[2.5.1. GreedyPlayer játékosprofil 9](#_Toc183455447)

[2.5.2. A\* játékosprofil 10](#_Toc183455448)

[2.6. Gépi tanulás 11](#_Toc183455449)

[2.7. Tesztelés 11](#_Toc183455450)

[2.8. Kiértékelés és eredmény analízis 11](#_Toc183455451)

[3. Rendszerterv 12](#_Toc183455452)

[3.1. Használati eset-diagram (Use-case UML) 12](#_Toc183455453)

[3.1.1. Használati esetek részletes szöveges ismertetése 12](#_Toc183455454)

[3.1.2. Aktorok részletes leírása 13](#_Toc183455455)

[3.2. Tevékenység-diagram (Action UML) 14](#_Toc183455456)

[3.3. Állapotgép diagramok (State Machine) 14](#_Toc183455457)

[3.4. Kontextus diagram (Context diagram) 14](#_Toc183455458)

[3.5. Osztály diagram (Class UML) 15](#_Toc183455459)

[3.6. Adatbázis egyedkapcsolat diagram (ER) 16](#_Toc183455460)

[4. Technológia 19](#_Toc183455461)

[4.1. Fejlesztői környezet 19](#_Toc183455462)

[4.1.1. Microsoft Visual Studio Code 19](#_Toc183455463)

[4.1.2. JetBrains PyCharm Professional: 19](#_Toc183455464)

[4.2. Paradigmák és implementációs szabványok 19](#_Toc183455465)

[4.3. Verziókövetés 19](#_Toc183455466)

[4.4. Programnyelvek 19](#_Toc183455467)

[4.4.1. Python 19](#_Toc183455468)

[4.5. Függvénykönyvtárak 19](#_Toc183455469)

[4.6. Futási környezet 19](#_Toc183455470)

[4.7. Egyéb felhasznált eszközök és szoftverek 20](#_Toc183455471)

[4.7.1. GanttProject 20](#_Toc183455472)

[4.7.2. Software Ideas Modeler 20](#_Toc183455473)

[4.7.3. Figma 20](#_Toc183455474)

[4.7.4. Inno Setup 20](#_Toc183455475)

[5. Mérföldkövek és feladatok 21](#_Toc183455476)

[5.1. Játék motor 21](#_Toc183455477)

[5.1.1. Logika 21](#_Toc183455478)

[5.1.2. Felhasználói interakció 21](#_Toc183455479)

[5.1.3. Párhuzamos mellékfeladatok 21](#_Toc183455480)

[5.2. Prototípus I. 21](#_Toc183455481)

[5.2.1. Mohó algoritmus 21](#_Toc183455482)

[5.2.2. Grafikus felhasználói felület v.0.1 21](#_Toc183455483)

[5.2.3. Testing I. 21](#_Toc183455484)

[5.3. Prototípus II. 22](#_Toc183455485)

[5.3.1. Grafikus felhasználói felület v.1.0 22](#_Toc183455486)

[5.3.2. Egyéb funkciók 22](#_Toc183455487)

[5.3.3. testing II. 22](#_Toc183455488)

[6. Projekttervezés 23](#_Toc183455489)

[6.1. Kockázatterv 23](#_Toc183455490)

[6.1.1. Kockázatelemzési bevezető 23](#_Toc183455491)

[6.1.2. Halszálka-diagram 23](#_Toc183455492)

[6.1.3. Pareto-diagram 24](#_Toc183455493)

[6.1.4. SWOT analízis 25](#_Toc183455494)

[6.2. Csapattagok leterheltsége 26](#_Toc183455495)

[6.3. Ütemterv (Gantt-diagram) 26](#_Toc183455496)

[6.4. Projektszervezési modell kiválasztása 28](#_Toc183455497)

[6.5. Tesztelési terv (Unittest) 28](#_Toc183455498)

[6.5.1. Teszt I. megközelítés 28](#_Toc183455499)

[6.5.2. Teszt II. megközelítés 29](#_Toc183455500)

[6.5.3. Teszt I. hipotézisei 29](#_Toc183455501)

[6.5.4. Teszt II. hipotézisei 30](#_Toc183455502)

[6.5.5. Teszt I. 30](#_Toc183455503)

[6.5.6. Teszt elemzés I. 31](#_Toc183455504)

[6.5.7. Teszt II. 31](#_Toc183455505)

[6.5.8. Teszt elemzés II. 32](#_Toc183455506)

[6.6. Projekt lezárása 32](#_Toc183455507)

[6.7. Projekt utóélete 32](#_Toc183455508)

[Összefoglalás 33](#_Toc183455509)

[Ábrajegyzék 34](#_Toc183455510)

[Melléklet 35](#_Toc183455511)

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
* 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 1: 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 a mohó-, és az A\* keresési algoritmusokat alkalmazzuk. 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.

* 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. 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)

ábra 4: A játék használati eset-diagramja

A képen szöveg, diagram, Betűtípus, sor látható

Automatikusan generált leírás

* + 1. Használati esetek részletes szöveges ismertetése

**Install Game**

A játékállományok telepítése saját rendszerre, melyhez először a játéktelepítő letöltése

szükséges.

**Play Game**

A játék futtatása, történhet **CvC** (**C**omputer **v**s **C**omputer), **PvC** (**P**layer **v**s **C**omputer) és **PvP** (**P**layer **v**s **P**layer) játékmódokban, valamint minden játékmódban tetszés szerint használható a **Log Game** eset.

**Access Scoreboard**

Hozzáférés a közös ranglétrához, ahol az összes egyéni legjobb eredmény elérhető és megtekinthető.

**Authenticate**

A játékos azonosításának esete, ami kötelezően maga után vonja a UserID validációt.

**Improve Game Engine**

A játékmotor fejlesztésének esete (kötelező dokumentációs folyamattal).

**Fix Bugs**

A hibák javításainak folyamata (kötelező dokumentációs folyamattal).

**Analyise Game Data**

A lejátszott mérkőzések kiértékelésének folyamata (kötelező dokumentációs folyamattal).

**Improve AI Profiles**

A gépi profilok továbbfejlesztésének esete (kötelező dokumentációs folyamattal).

**Design Interface**

A felhasználó felület tervezésének esete (kötelező dokumentációs folyamattal).

**Draw Mockup**

A tervezést követő látványtervek elkészítése, megrajzolása (kötelező dokumentációs folyamattal).

**Implement codebase**

A felhasználói felület implementációja a tervezet, valamint a kidolgozott látványtervek alapján (kötelező dokumentációs folyamattal).

**Write test**

A játék állományainak automatizált tesztelése, egyes esetekben moduláris megközelítésben máskor integrációs jelleggel (kötelező dokumentációs folyamattal).

* + 1. Aktorok részletes leírása

**New Player**

Új játékos, aki még nem telepítette a játékot.

**Player**

A játékkal már rendelkező általános játékos.

**Engine Architect**

A játékmotor fejlesztéséért és javításáért felelős szakember.

**AI Engineer (and Data Analyst)**

A gépi játékos profilok fejlesztéséért és a meccsadatok kiértékeléséért felelős fejlesztő / adatelemző.

**UI Designer**

A felhasználói felület tervezéséért is implementálásáért felelős szakember.

**Tester**

A kódbázis és a funkciók teszteléséért felelős szakember.

* 1. Tevékenység-diagram (Action UML)

Megegyezik a használati eset diagrammal.

* 1. 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 5: A játék teljes felhasználói állapotgép diagramja

* 1. Kontextus diagram (Context diagram)

A white square with black text

Description automatically generated

ábra 6: 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)

A Player osztály konstruktora egy color paramétert fogad el, amely a játékos színét jelzi. Az abstractmethod dekorátorral ellátott „make\_move()” metódus azt jelenti, hogy az összes leszármazott osztálynak implementálnia kell ezt a lépési logikát. A „choose\_opponent\_piece\_to\_remove()” statikus metódus egy eltávolítandó ellenfél bábut választ, de a konkrét logikát még nem tartalmazza. Az osztály tartalmaz egy metódust is, amely a játékos nevét és színét adja vissza szöveges formában. (4.ábra)

A screenshot of a computer

Description automatically generated

ábra 7: Player osztály és öröklődés

* 1. Adatbázis egyedkapcsolat diagram (ER)

A képen szöveg, sor, diagram, képernyőkép látható

Automatikusan generált leírás

ábra 8: Adatbázis egyedkapcsolat diagram (ER)

Egyedkapcsolat diagram leírása:

1. Players Tábla (Játékosok)

A players tábla tárolja a játékosokat és az azokkal kapcsolatos információkat.

|  |  |  |
| --- | --- | --- |
| Mezők neve | Típus | Megjegyzés |
| player\_id | int | A játékos egyedi azonosítója (elsődleges kulcs, automatikusan növekvő érték) |
| color | varchar(1) | A játékos színe: 'W' (Fehér) vagy 'B' (Fekete) |
| name | varchar(50) | A játékos neve, például "GreedyPlayer", "MLPlayer" |

* **player\_id**: Egyedileg azonosítja a játékost. Az érték automatikusan növekvő, tehát minden új játékosnak egyedi azonosítót rendel.
* **color**: Az osztály színe, ami meghatározza, hogy melyik félhez tartozik a játékos.
* **name**: A játékos neve, amit a program különböző típusú játékosok, mint például GreedyPlayer, MLPlayer, stb., esetében adhat meg.

**2. Games Tábla (Játékok)**

A games tábla tárolja az összes játékot és annak aktuális állapotát.

|  |  |  |
| --- | --- | --- |
| **Mezők neve** | **Típus** | **Megjegyzés** |
| game\_id | int | A játék egyedi azonosítója (elsődleges kulcs, automatikusan növekvő érték) |
| board\_state | text | A játék állapota, például egy sztringben tárolva (JSON vagy egyéb formátum) |
| current\_turn | int | Az aktuális játékos, aki következőként léphet (a players.player\_id hivatkozása) |

* **game\_id**: Az egyes játékok egyedi azonosítója, amely az adatbázisban való nyilvántartást biztosítja.
* **board\_state**: A játék táblájának állapota. Ez tárolhatja a táblát szöveges formában, például JSON-ban vagy egyéb formátumban.
* **current\_turn**: Az aktuális játékos, akinek a következő lépése jön. A players.player\_id mezőhöz kapcsolódik.

**3. Moves Tábla (Lépések)**

A moves tábla tárolja az összes lépést, amelyet egy játékos végrehajt a játékmenet során.

|  |  |  |
| --- | --- | --- |
| **Mezők neve** | **Típus** | **Megjegyzés** |
| move\_id | int | A lépés egyedi azonosítója (elsődleges kulcs, automatikusan növekvő érték) |
| game\_id | int | Az a játék, amelyhez a lépés tartozik (a games.game\_id hivatkozása) |
| player\_id | int | Az a játékos, aki a lépést tette (a players.player\_id hivatkozása) |
| start\_position | int | A lépés kezdő pozíciója. |
| end\_position | int | A lépés befejező pozíciója. |

* **move\_id**: Az egyedi azonosítója minden lépésnek.
* **game\_id**: A játék azonosítója, amelyhez a lépés tartozik. Ez a games.game\_id mezőhöz kapcsolódik.
* **player\_id**: A játékos azonosítója, aki végrehajtotta a lépést. Ez a players.player\_id mezőhöz kapcsolódik.
* **start\_position**: A lépés kezdő pozíciója.
* **end\_position**: A lépés befejező pozíciója.

**4. Removed Pieces Tábla (Eltávolított Korongok)**

A removed\_pieces tábla tárolja azokat a korongokat, amelyeket egy játékos eltávolított a játékról.

|  |  |  |
| --- | --- | --- |
| **Mezők neve** | **Típus** | **Megjegyzés** |
| piece\_id | int | Az eltávolított korong egyedi azonosítója (elsődleges kulcs, automatikusan növekvő érték) |
| game\_id | int | Az a játék, amelyhez az eltávolított korong tartozik (a games.game\_id hivatkozása) |
| player\_id | int | Az a játékos, aki eltávolította a korongot (a players.player\_id hivatkozása) |
| piece\_position | int | Az eltávolított korong pozíciója a táblán (pl. egy numerikus pozíció) |

* **piece\_id**: Az eltávolított korong egyedi azonosítója.
* **game\_id**: Az a játék, amelyhez a korong eltávolítása tartozik. Ez a games.game\_id mezőhöz kapcsolódik.
* **player\_id**: Az a játékos, aki eltávolította a korongot. Ez a players.player\_id mezőhöz kapcsolódik.
* **piece\_position**: Az eltávolított korong pozíciója a táblán.

**5. Relációk (Kapcsolatok)**

A relációk a különböző táblák között biztosítják az összefüggéseket és az integritást.

* **games.current\_turn > players.player\_id**:
  + Ez a kapcsolat azt jelenti, hogy minden játékhoz hozzárendelünk egy játékost, akinek a következő lépését kell végrehajtania. A current\_turn mező a players.player\_id mezőre hivatkozik.
* **moves.game\_id > games.game\_id**:
  + Ez a kapcsolat biztosítja, hogy minden egyes lépés egy adott játékhoz tartozik. A moves.game\_id hivatkozik a games.game\_id mezőre.
* **moves.player\_id > players.player\_id**:
  + Mivel minden lépést egy játékos hajt végre, a moves.player\_id mező a players.player\_id mezőre hivatkozik, így kapcsolva össze a játékosokat a lépésekkel.
* **removed\_pieces.game\_id > games.game\_id**:
  + Mivel minden eltávolított korong egy játékhoz tartozik, a removed\_pieces.game\_id hivatkozik a games.game\_id mezőre.
* **removed\_pieces.player\_id > players.player\_id**:
  + Az eltávolított korongok egy játékoshoz kapcsolódnak, így a removed\_pieces.player\_id hivatkozik a players.player\_id mezőre.

**Összegzés:**

A fenti relációk segítségével az adatbázis biztosítja a játékosok, játékok, lépések és eltávolított korongok közötti összefüggéseket. Az adatbázis sémát úgy alakítottuk ki, hogy biztosítsa a játék állapotának nyomon követését és a különböző események (lépések, eltávolítások) helyes kezelését.

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

* Verzió: 1.94.2
* Build: 10262041
* 2024.09
  + 1. JetBrains PyCharm Professional:
* 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
* 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
* Verzió: 124.4.7
* 2024.09.18.
  + 1. Inno Setup
* Verzió: 6.3.3
* 2024. 11. 23.

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. Mohó 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. 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. Projekttervezés
   1. Kockázatterv
      1. Kockázatelemzési bevezető

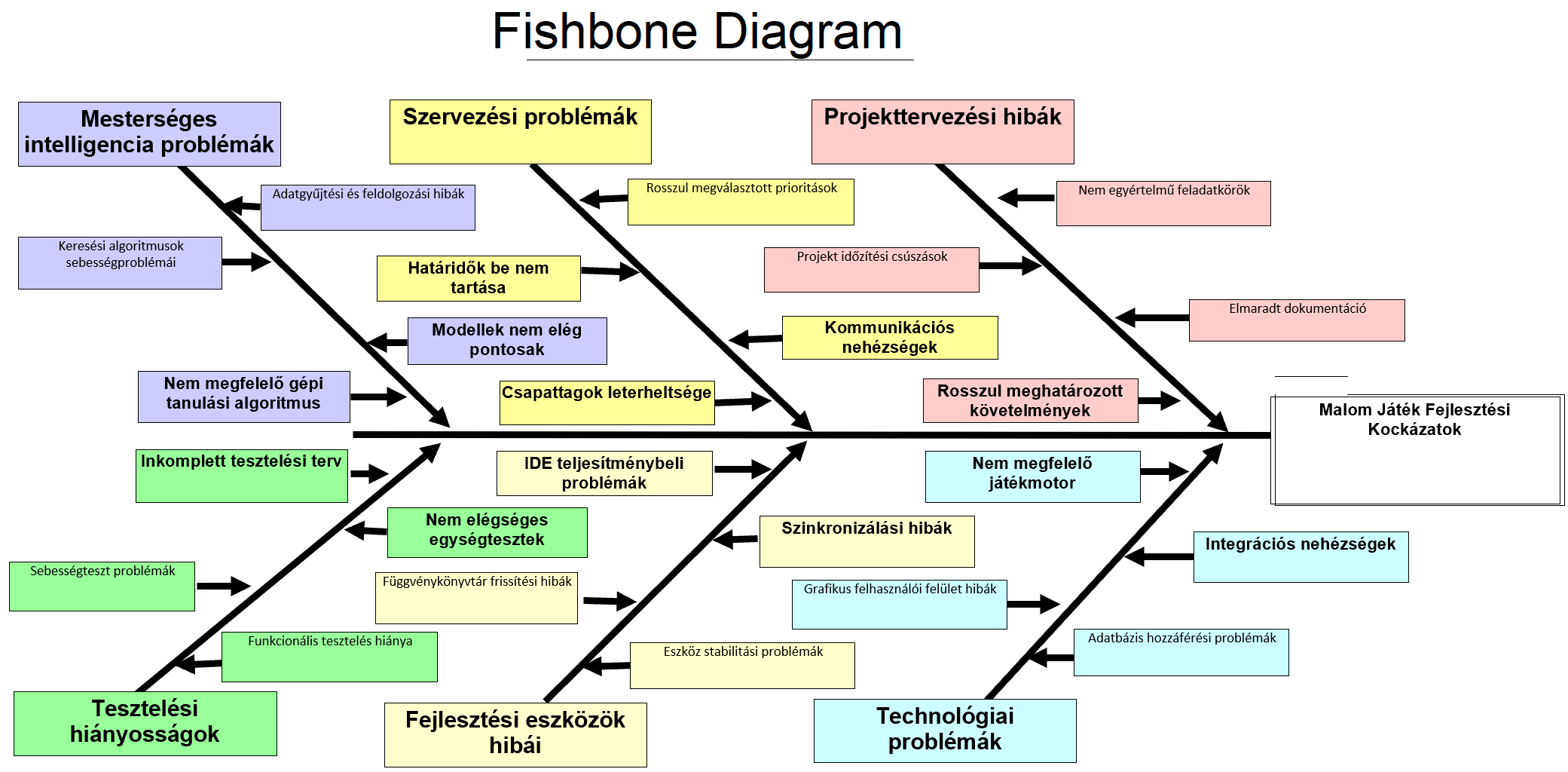
A három különböző tantárgyra készített projekt jelentős kockázatot hordoz, mivel a követelmények összehangolása, a különböző elvárások kielégítése és a megnövekedett időigény jelentős tervezési és fejlesztési többletet eredményez. A siker kulcsa a részletes előzetes tervezés, a követelmények pontos megértése 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 9: 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 10: 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 11: 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 12: 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 13: 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 14: 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 15: 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 16: Gantt-diagram a negyedik mérföldkőig

* 1. Projektszervezési modell kiválasztása

A projekt számára az agilis megközelítésen belül a Scrum modellt választottuk, mivel ez a módszertan kiválóan támogatja az iteratív és inkrementális fejlesztést, valamint gyors alkalmazkodást tesz lehetővé a változó követelményekhez. A Scrum módszertan lehetővé teszi, hogy a csapat rövid, kéthetes sprintekben dolgozzon, és ezen időszakok végén egy új stabil verziót, vagyis "build"-et szállítson, amely már tartalmazza a legfrissebb fejlesztéseket. A projekt során különös hangsúlyt helyezünk az egyszerűségre, a letisztultságra és az átláthatóságra, melyek mind a sikeres termék fejlesztésének alapvető feltételei.

A csapat rendszeres standup meetingeken vesz részt, amelyek két hetente az egész csapatot érintik, és lehetőséget biztosítanak a projekt aktuális állapotának összefoglalására, a problémák és akadályok azonosítására, valamint a célok közös áttekintésére. Az alcsapatok – például a motorfejlesztés, felhasználói felület (GUI), mesterséges intelligencia (AI), illetve a játék egészének fejlesztése területén – saját ütemezésük szerint is szervezhetnek megbeszéléseket, amely rugalmasságot biztosít számukra a hatékony együttműködéshez.

A fejlesztés folyamán a mérce a működő szoftver, vagyis a hangsúly azon van, hogy a létrehozott funkciók valós értéket képviseljenek és megbízhatóan működjenek. A projektmenedzserrel való rendszeres visszacsatolások során a csapat tagjai az elért eredményekről és a felmerült problémákról egyeztetnek, ezzel biztosítva, hogy a projekt a megfelelő irányba haladjon. A fejlesztési folyamatban jelentős szerepet kapnak a párhuzamosított tevékenységek, melyek lehetővé teszik az egyes területek egymástól független előrehaladását, így gyorsabb fejlesztést és hatékonyabb munkamegosztást tesznek lehetővé.

* 1. Tesztelési terv (Unittest)
     1. Teszt I. megközelítés

A tesztelési terv célja annak biztosítása, hogy a játékmenet logikája megfelelően működjön, és hogy minden játékrendszerre vonatkozó feltétel helyesen legyen kezelve. A tesztek az egyes játékköröket és a különböző játékállapotokat fogják ellenőrizni, például a játékosok közötti váltást, a bábuelhelyezést, a mozgást, a bábuk eltávolítását és a játék végi feltételeket.

A tesztelési megközelítés során az alábbiakat vizsgáljuk:

* A játék alapállapotainak helyessége a játék indításakor.
* A játékosok közötti váltások megfelelő működése.
* A mozgás regisztrálása, beleértve a bábu elhelyezést és a mozgást.
* Az ellenfél bábujának eltávolítása és annak helyes működése.
* A játék végállapota és a játék végi feltételek ellenőrzése.
* A lehetséges érvényes lépések generálása és annak tesztelése.

A tesztelés során nemcsak a logikai hibák, hanem az esetleges szintaktikai vagy mechanikai problémák is észlelhetők, így az alkalmazás minden aspektusát megfelelően ellenőrizni lehet.

* + 1. Teszt II. megközelítés

A tesztelési terv célja annak biztosítása, hogy a GUI komponensek és azok működése a várt módon teljesítenek. Az alkalmazás Tkinter és customtkinter alapú, így a tesztelés során figyelembe kell venni a GUI elemek interakcióját és a felhasználói élményt. A tesztelés során minden egyes GUI műveletet külön-külön ellenőrzünk, hogy biztosak legyünk abban, hogy az alkalmazás helyesen működik, és hogy a GUI elemek megfelelően reagálnak a különböző beállításokra, például megjelenési módok, képek betöltése, játékmenü és alkalmazás bezárás.

A tesztelési megközelítés során az alábbiakat vizsgáljuk:

* A GUI elemek inicializálása és azok helyes alapértékei.
* A képek betöltése és azok típusainak helyessége.
* A GUI megjelenítése és ablakainak helyes indítása.
* Az alkalmazás megjelenésének és témájának frissítése.
* A játékmenü betöltése és annak vizuális elemei.
* Az alkalmazás megfelelő leállítása.

A tesztelés során mockolás segítségével elkerüljük a külső függőségek (mint például a játékmenet vagy az eseménykezelő) tényleges inicializálását, ezáltal gyorsabbá és könnyebbé téve a tesztelést.

* + 1. Teszt I. hipotézisei

A tesztelés alapját az alábbi hipotézisek képezik:

* A játék indításakor az alapállapotok helyesen kerülnek beállításra (kezdő játékos, elhelyezett bábuk száma, táblaméret).
* A játékosok közötti váltás minden lépésnél megfelelően történik.
* A bábuk helyes regisztrálása történik, és a táblán való elhelyezés is sikeres.
* A játékosok mozgása és az ellenfél bábujának eltávolítása hibátlanul működik.
* A játék végi feltételek (például, ha egy játékosnak kevesebb mint 3 bábúja marad) helyesen kerülnek ellenőrzésre.
* A generált érvényes lépések megfelelnek a játékszabályoknak, és helyes lépéseket biztosítanak a játékosok számára.
  + 1. Teszt II. hipotézisei

A tesztelés alapját az alábbi hipotézisek képezik:

* A GUI inicializálásakor a megfelelő alapértékek kerülnek beállításra.
* A képek sikeresen betöltődnek és megfelelő típusú objektumokként jelennek meg.
* A GUI indítása során a megfelelő ablakot hozza létre.
* A megjelenési mód és téma frissítése működik, és az alkalmazás a várt módon változik.
* A játékmenü betöltésekor a megfelelő widgetek jelennek meg.
* Az alkalmazás leállítása megfelelően végrehajtódik, és a rendszer kiemeli a kilépési folyamatot.
  + 1. Teszt I.

A teszt során a következő lépéseket követjük:

* test\_initial\_conditions: Ellenőrizzük, hogy a játék alapállapotai megfelelően vannak beállítva az indításkor. Ez magában foglalja a kezdő játékost, a bábuk számát és a tábla méretét.
* test\_switch\_turns: Teszteljük, hogy a játékosok közötti váltás megfelelően működik, és hogy a switch\_turns metódus sikeresen váltja a játékosokat.
* test\_register\_move\_placing: Teszteljük, hogy a bábu elhelyezését helyesen regisztrálja a játék, és hogy az első játékos helyes pozícióban helyezi el a bábuját.
* test\_register\_move\_moving: Ellenőrizzük, hogy a bábu mozgása helyesen történik meg, és hogy a játékosok közötti váltás után az első játékos helyesen mozgatja a bábuját.
* test\_remove\_opponent\_piece: Teszteljük, hogy az ellenfél bábujának eltávolítása megfelelően történik, és hogy a tábla frissül a megfelelő helyeken.
* test\_game\_over\_conditions: Teszteljük a játék végének feltételeit, beleértve azt is, amikor egy játékosnak kevesebb mint három bábúja marad.
* test\_generate\_valid\_moves: Ellenőrizzük, hogy a generate\_valid\_moves metódus megfelelően generálja a valid lépéseket, és hogy a szabályoknak megfelelően kerülnek felajánlásra a lehetséges lépések.

A tesztek célja annak biztosítása, hogy a játék minden szabálya és feltétele hibátlanul működjön, a játékosok mozgása, a bábu elhelyezése, és a játék végi feltételek is megfelelően kezelődnek.

* + 1. Teszt elemzés I.

A tesztelés elemzése során az alábbiakra koncentrálunk:

* Teszt lefutása: Minden tesztet külön-külön futtatunk, hogy biztosítsuk, hogy minden egyes játékmechanizmus helyesen működik. A tesztek gyorsan ellenőrzik az összes kulcsfontosságú műveletet, mint például a játékosok közötti váltás, a bábu elhelyezése, a mozgás regisztrálása, a bábuk eltávolítása, valamint a játékvégi feltételek.
* Hibák és problémák: A hibák és problémák a tesztelési eredményekben fognak megjelenni, például, ha egy játékos nem tud lépni, vagy ha a játékvégi feltételek nem kerülnek megfelelően észlelésre.
* Kód javítása: A tesztelési hibák segítenek azonosítani a rendszerben található hibákat, és segítenek azok gyors javításában. A hibák kijavítása után az érintett teszteket újra futtatjuk a helyes működés biztosítása érdekében.
* Tesztkörnyezet: A tesztelés során használt környezetet, beleértve a Python verzióját és a használt játéklogikát, dokumentálni kell, hogy minden teszt újraindítása során biztosíthassuk a kompatibilitást.
  + 1. Teszt II.

A teszt során a következő lépéseket követjük:

* test\_initial\_values: Ellenőrizzük, hogy az inicializáláskor a megfelelő alapértékek kerülnek beállításra. Ez magában foglalja a játékmódot, a nehézségi szintet, az AI típusát, és a naplózási beállítást.
* test\_load\_images: Teszteljük, hogy a képek sikeresen betöltődnek, és hogy azok megfelelő típusú objektumok (például ImageTk.PhotoImage).
* test\_start\_gui: Megvizsgáljuk, hogy a GUI indítása elindítja a megfelelő ablakot, és hogy az alkalmazás indítása sikeresen megtörténik.
* test\_update\_appearance: Teszteljük, hogy a megjelenési mód változása helyesen frissíti a GUI-t. Az update\_appearance metódusnak a customtkinter set\_appearance\_mode függvényét kell hívnia a megfelelő paraméterekkel.
* test\_update\_theme: Ellenőrizzük, hogy a téma frissítése a kiválasztott szín alapján történik, és hogy az alapértelmezett szín beállításai helyesen működnek.
* test\_show\_game\_settings\_menu: Teszteljük, hogy a játékmenü megjelenítésekor a megfelelő widgetek töltődnek be.
* test\_exit\_app: Ellenőrizzük, hogy az alkalmazás leállítása a SystemExit kivételt dobja, biztosítva a megfelelő kilépést.

A fent említett tesztek biztosítják, hogy a GUI minden funkciója megfelelően működik, és a felhasználói interakciók során nincsenek hibák.

* + 1. Teszt elemzés II.

A tesztelés elemzése során az alábbiakra koncentrálunk:

* Teszt lefutása: Minden egyes tesztet külön-külön futtatunk, hogy ellenőrizzük, hogy a rendszer minden funkciója megfelelően működik. Mivel a tesztek mockolás segítségével működnek, így az alapvető GUI műveletek tesztelése mellett nem szükséges valós felhasználói interakciókat szimulálni.
* Hibák és problémák: Az esetleges hibák és problémák a tesztelési eredményekben fognak megjelenni, például, ha a képek nem töltődnek be, vagy ha egy widget nem jelenik meg a várt módon.
* Kód javítása: Amennyiben bármelyik teszt nem fut le sikeresen, a hibák gyorsan lokalizálhatók, és a megfelelő kódjavítást követően újraindíthatóak a tesztek, hogy biztosítsák a működést.
* Tesztkörnyezet: A tesztelés során használt környezetet (például a Tkinter és customtkinter verzióit, valamint a Python verzióját) dokumentálni kell, hogy minden teszt újraindítása során biztosítani lehessen a kompatibilitást.
  1. Projektzárás és szoftverutóélet

A tervezett funkciók és játékmódok sikeresen elkészültek, a játék teljesértékűen működik mind debug mind grafikai módban. A könnyű telepítés érdekében külön telepítő szoftver is rendelkezésre áll a felhasználók számára. A kódbázis moduláris felépítésének köszönhetően a jövőben további gépi profilok kidolgozása és implementálása is könnyen kivitelezhető. A projekt szabadon elérhető a GitHubon a *Creative Commons Attribution-NonCommercial (CC BY-NC)* licensz megkötései szerint, vagyis szabadon másolható és módosítható (fejleszthető) viszont nem használható anyagi haszonszerzésre.

Beszámoló

Kihívások, eredmények, fejlesztési lehetőségek

Ábrajegyzék

[ábra 1: Megjelenítő felület kezdeti és kiforrottabb koncepciós vázlata 8](#_Toc183454233)

[ábra 2: GreedyPlayer algoritmikus felépítése (Flow Chart) 10](#_Toc183454234)

[ábra 3: A\* algoritmus heurisztikájául szolgáló döntési fa 10](#_Toc183454235)

[ábra 4: A játék használati eset-diagramja 12](#_Toc183454236)

[ábra 5: A játék teljes felhasználói állapotgép diagramja 14](#_Toc183454237)

[ábra 6: A malomjáték szoftver kontextus diagramja 14](#_Toc183454238)

[ábra 7: Player osztály és öröklődés 15](#_Toc183454239)

[ábra 8: Adatbázis egyedkapcsolat diagram (ER) 16](#_Toc183454240)

[ábra 9: Malom Játék Fejlesztési Kockázatainak Halszálka-diagramja 24](#_Toc183454241)

[ábra 10: Pareto-diagram a főbb kockázatokról 25](#_Toc183454242)

[ábra 11: A FreeKredit fejlesztőcsapat SWOT elemzése 25](#_Toc183454243)

[ábra 12: Fejlesztői csapat erőforrás-diagramja 26](#_Toc183454244)

[ábra 13: Gantt-diagram az első mérföldkőig 26](#_Toc183454245)

[ábra 14: Gantt-diagram a második mérföldkőig 27](#_Toc183454246)

[ábra 15: Gantt-diagram a harmadik mérföldkőig 27](#_Toc183454247)

[ábra 16: Gantt-diagram a negyedik mérföldkőig 27](#_Toc183454248)

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