A képen szöveg látható

Automatikusan generált leírás

Rush Hour játékfejlesztés

Rush hour game development

Szabó Dániel

Z3IK64

Dr. Bolla Kálmán, főiskolai docens

2023

Tartalomjegyzék

[1. Bevezetés 8](#_Toc177305918)

[1.1. A szakdolgozat célja 8](#_Toc177305919)

[1.2. Felhasznált algoritmusok és eszközök 8](#_Toc177305920)

[1.3. A szakdolgozat felépítése 8](#_Toc177305921)

[2. A feladat elemzése, a probléma érintő bemutatása (3 oldal) 9](#_Toc177305922)

[2.1. A játék szabályai 9](#_Toc177305923)

[2.2. A megoldandó problémák 9](#_Toc177305924)

[2.2.1. Generáló algoritmus 9](#_Toc177305925)

[2.2.2. Megoldó algoritmus 9](#_Toc177305926)

[2.3. Technológiák 9](#_Toc177305927)

[2.3.1. Unity 9](#_Toc177305928)

[2.3.2. Ha lesz adatbázis akkor azt ide fogom leírni (talán sqlite lesz vagy mariadb) 10](#_Toc177305929)

[3. Szoftver tervezése (12 oldal) 11](#_Toc177305930)

[3.1. Funkciók 11](#_Toc177305931)

[3.1.1. Játék nehézségének kiválasztása 11](#_Toc177305932)

[3.1.2. Járművek és pálya megjelenítése 11](#_Toc177305933)

[3.1.3. Feladvány generálása 11](#_Toc177305934)

[3.1.4. Megoldó algoritmus 11](#_Toc177305935)

[3.1.5. Járművek mozgathatósága a pályán 11](#_Toc177305936)

[3.1.6. Játék ciklus 11](#_Toc177305937)

[3.1.7. Eltárolt adatok (pontszám,idő) [adatbázis????] 11](#_Toc177305938)

[3.1.8. Játékot színesítő elemek (zene,hangok,mozgó járművek a porkolón kívül) 11](#_Toc177305939)

[3.2. Adatszerkezetek, objektumok, interfészek (UML) 11](#_Toc177305940)

[3.3. Tesztelés 11](#_Toc177305941)

[3.3.1. Algoritmusok tesztelése 11](#_Toc177305942)

[3.3.2. Megjelenítés tesztelése 11](#_Toc177305943)

[3.3.3. Manuális tesztelés 11](#_Toc177305944)

[4. Szoftver kialakítása (24 oldal) 12](#_Toc177305945)

[4.1. Menü 12](#_Toc177305946)

[4.1.1. Főmenü 12](#_Toc177305947)

[4.1.2. Statisztika (ha lesz adatbázis) 12](#_Toc177305948)

[4.1.3. Beállítások 12](#_Toc177305949)

[4.2. Játék 12](#_Toc177305950)

[4.2.1. Felület 12](#_Toc177305951)

[4.2.2. Interakciók (Autók és kamera mozgatása) 12](#_Toc177305952)

[5. Tesztelés, módosítások (12 oldal) 13](#_Toc177305953)

[5.1.1. Pálya generálása 13](#_Toc177305954)

[5.1.2. Járművek kezelése (lehelyezés a táblára, mozgatás) 13](#_Toc177305955)

[5.1.3. Előretekintő keresés 13](#_Toc177305956)

[5.1.4. Gameobject-ek létrehozása 13](#_Toc177305957)

[5.1.5. Játék ciklus 13](#_Toc177305958)

[6. Összefoglalás (1 oldal) 14](#_Toc177305959)

[Idegen nyelvű összefoglalás (1 oldal) 15](#_Toc177305960)

[Ábrajegyzék 16](#_Toc177305961)

[Irodalomjegyzék 17](#_Toc177305962)

[Melléklet 18](#_Toc177305963)

1. Bevezetés
   1. A szakdolgozat célja

A szakdolgozat célja a Rush Hour játék elkészítése érdekes és skálázható nehézségű feladványokkal. A játék lényege, hogy az autónkat kijuttassuk egy járművekkel teli parkolóból. A játékot fizikai és webes formában is megvalósították. Ezek a változatok nem véletlenszerű pályákat kínálnak, hanem előre megtervezett és fix nehézségű feladványokat, ami korlátozza a hosszú távú újrajátszhatóságot.

Azért döntöttem a feladat mellett, mert mindig érdekelt a játékfejlesztés és a program nehézségei érdekesek és megfelelő kihívást jelent a szakdolgozathoz.

* 1. Felhasznált algoritmusok és eszközök

A dolgozatban egy olyan megoldás kidolgozására törekszem, amely lehetőséget ad a nehézség kiválasztására és véletlenszerű pályák generálására mesterséges intelligencia alapú algoritmusok segítségével, mint a visszalépéses keresés és a heurisztikus előretekintő keresés. A választásom azért esett erre a megoldásra, mert ezek az algoritmusok könnyen alkalmazhatók és optimalizálhatók az adott problémára, valamint különösen hatékonynak bizonyultak hasonló játékok feladványainak létrehozásában. Azért is részesítettem előnyben ezeket az algoritmusokat, mivel korábbi próbálkozásaim nem vezettek eredményre. Az első megközelítés során a megoldás összekeverésével érte volna el véletlenszerű pályákat, ezek viszont nem voltak skálázhatók és könnyen megoldhatók voltak.

A játék motorjának a Unity Engine-t választottam a könnyű használat miatt és a későbbi továbbfejlesztési lehetőségekért a felhasználói igényeknek megfelelően.

* 1. A szakdolgozat felépítése

A dolgozat során részletesen bemutatom a pályagenerálás folyamatát, a nehézség meghatározását, a megoldó algoritmust és a játék apróbb részleteit, mint az interakció a járművekkel, a környezet és a játékciklus.

1. A feladat elemzése, a probléma érintő bemutatása (3 oldal)
   1. A játék szabályai

Az alap játék egy 6x6-os rácson játszódik, ahol autók és kamionok helyezkednek el. A feladat, hogy a saját autónkat kijuttassuk a parkolóból, viszont más járművek ezt megakadályozzák. A járművek elhelyezkedhetnek vízszintesen és függőlegesen és csak ezen a tengelyen mozgathatók, addig ameddig egy másik jármű nem állja az útjukat.

* 1. A megoldandó problémák
     1. Generáló algoritmus

Az generáló algoritmus visszalépéses keresést használ. Egy üres pálya legenerálásával kezdődik, ahol minden mező értéke nulla. Ezután az egyes mezőkre letehető összes lehetséges jármű információját eltárolja. Miután lehelyez egy járművet lefut egy megoldó algoritmus, ami ellenőrzi, hogy megoldható-e a feladvány, ha megoldható akkor az összes letehető járművek listájából törlésre kerülnek a megfelelő elemek a játék szabályai alapján, ha nem megoldható a feladvány, akkor törlésre kerül a jármű a pályáról és újat helyez le.

* + 1. Megoldó algoritmus

A megoldó algoritmus előretekintő kereséssel a jelenlegi állapotból minden jármű összes lehetséges mozgatásából generál további állapotokat. Heurisztika alapján kiszámolja az egyes állapotok értékét és a legkisebb értékű állapotot fejti ki, ameddig nem találja meg a megoldást vagy eléri a száz lépést.

* 1. Technológiák
     1. Unity

A játék motorjának a Unity-t használtam, ami támogatja a C# nyelvet. Elsősorban az érthető és részletes dokumentációja miatt választottam, valamint az ingyenes boltból (Asset Store) könnyen hozzáférjek és integráljak modelleket a játékba. Egyik erőssége más motorokhoz képest, hogy rengeteg előre létrehozott függvényt tartalmaz, ami megkönnyíti a fejlesztést. Többek között mátrix transzformációs függvényeket és beépített fizikai motort is tartalmaz. A fejlesztés alatt és a végleges játékban könnyen lehet teszteket írni és futtatni a Unity Test Framework segítségével. Felhőalapú szinkronizálást és verziókövetést is biztosít.

* + 1. Ha lesz adatbázis akkor azt ide fogom leírni (talán sqlite lesz vagy mariadb)

valahova ide majd

eltárolnám a játékos nevét, legjobb idejét és mozgatásait az egyes nehézségeken

1. Szoftver tervezése (12 oldal)
   1. Funkciók
      1. Játék nehézségének kiválasztása

A fizikai Rush Hour játékban öt nehézségi szintet különböztet meg, amelyek a kezdő, gyakorlott, haladó, profi és nagymester.

A szakdolgozatban elkészített játékban is ezeket a szinteket különböztettem meg, attól függően, hogy a feladványt hány lépésben lehet megoldani.

A következő felosztásokat választottam a szintekhez:

* Kezdő szint: 10 - 20 lépés
* Gyakorlott szint: 21 – 45 lépés
* Haladó szint: 46- 65 lépés
* Profi szint: 66 – 80 lépés
* Nagymester szint: 81- 93 lépés

(itt szeretnék hivatkozni egy olyan kutatásra, aminek az lett az eredménye, hogy maximum 93 lépésből bármelyik feladvány megoldható.)

(leírnám hogy az algoritmusban, hogyan jelenik meg a nehézség és a lépések számának korlátozása szinthez függően.)

(bemutatnám a tervet arra hogy ez a menü hogy néz ki, és leírnám hogyan épül fel)

* + 1. Járművek és pálya megjelenítése

(Place, Board osztály leírása)

A dolgozat során a logika és a megjelenítés szétválasztására törekedtem, az olvashatóbb kód, a könnyebb tesztelési lehetőségek és a javíthatóság növelése érdekében, ezért a megjelenítés kódja akkor fut le amikor már a logikai részek lefutottak.

A pályát egy 2 dimenziós tömbként hozza létre a játék, ami a pálya méretétől függően feltölti az

összes értéket nullával.A pálya létrehozása mellett létrejön egy olyan lista is ami a 2 dimenziós tömb minden értékére létrehozza az oda letehető járművek tulajdonságait.

* + 1. 2.8.3. Járművek és pálya megjelenítése

(SpawnGrid,SpawnVehicle osztály leírása)

* + 1. 2.8.4. Feladvány generálása

(Puzzle generátor osztály leírása)

* + 1. 2.8.5. Megoldó algoritmus

(Solver osztály leírása)

* + 1. 2.8.6. Járművek mozgathatósága a pályán

(VehicleMovement osztály leírása)

* + 1. Játék ciklus
    2. Eltárolt adatok (pontszám,idő) [adatbázis????]
    3. Játékot színesítő elemek (zene,hangok,mozgó járművek a porkolón kívül)
  1. Adatszerkezetek, objektumok, interfészek (UML)

A

* 1. Tesztelés
     1. Algoritmusok tesztelése
     2. Megjelenítés tesztelése
     3. Manuális tesztelés

1. Szoftver kialakítása (24 oldal)

ez lesz az érdekes rész

* 1. Menü
     1. Főmenü
     2. Statisztika (ha lesz adatbázis)
     3. Beállítások
  2. Játék
     1. Felület
     2. Interakciók (Autók és kamera mozgatása)

1. Tesztelés, módosítások (12 oldal)

☺

Ez nem végleges kép



1.ábra: Tesztelések

* + 1. Pálya generálása
    2. Járművek kezelése (lehelyezés a táblára, mozgatás)
    3. Előretekintő keresés
    4. Gameobject-ek létrehozása
    5. Játék ciklus

1. Összefoglalás (1 oldal)

osszefoglalaaaas

Idegen nyelvű összefoglalás (1 oldal)

szia

Ábrajegyzék

[1.ábra: Tesztelések 12](#_Toc177303043)

Irodalomjegyzék

Melléklet