A képen szöveg látható

Automatikusan generált leírás

Csúcsforgalom játékfejlesztés

Rush hour game development

Szabó Dániel

Z3IK64

Dr. Bolla Kálmán Milán, főiskolai docens

2024

Tartalomjegyzék

[1. Bevezetés 8](#_Toc179717958)

[1.1. A szakdolgozat célja 8](#_Toc179717959)

[1.2. Felhasznált algoritmusok és eszközök 8](#_Toc179717960)

[1.3. A szakdolgozat felépítése 8](#_Toc179717961)

[2. A feladat elemzése, a probléma érintő bemutatása (3 oldal) 9](#_Toc179717962)

[2.1. A játék szabályai 9](#_Toc179717963)

[2.2. A megoldandó problémák 9](#_Toc179717964)

[2.2.1. Generáló algoritmus 9](#_Toc179717965)

[2.2.2. Megoldó algoritmus 9](#_Toc179717966)

[2.3. Technológiák 9](#_Toc179717967)

[2.3.1. Unity 9](#_Toc179717968)

[2.3.2. Adatbázis 10](#_Toc179717969)

[3. Szoftver tervezése (12 oldal) 11](#_Toc179717970)

[3.1. Funkciók 11](#_Toc179717971)

[3.1.1. Játék nehézségének kiválasztása 11](#_Toc179717972)

[3.1.2. Játék ciklus 12](#_Toc179717973)

[3.1.3. Regisztráció 12](#_Toc179717974)

[3.1.4. Bejelentkezés 13](#_Toc179717975)

[3.1.5. Ranglista 14](#_Toc179717976)

[3.1.6. Beállítások 15](#_Toc179717977)

[3.1.7. Zene 15](#_Toc179717978)

[3.1.8. Környezeti elemek 15](#_Toc179717979)

[3.2. Algoritmusok működése 16](#_Toc179717980)

[3.2.1. Generáló algoritmus 16](#_Toc179717981)

[3.2.2. Megoldó algoritmus 16](#_Toc179717982)

[3.2.3. Járművek és pálya megjelenítése 17](#_Toc179717983)

[3.2.4. Járművek és pálya megjelenítése 17](#_Toc179717984)

[3.3. Adatbázis 17](#_Toc179717985)

[3.3.1. Adatbázis terv 17](#_Toc179717986)

[3.4. Tesztelés 18](#_Toc179717987)

[3.4.1. Algoritmusok tesztelése 18](#_Toc179717988)

[3.4.2. Megjelenítés tesztelése 18](#_Toc179717989)

[3.4.3. Manuális tesztelés 18](#_Toc179717990)

[4. Szoftver kialakítása (24 oldal) 19](#_Toc179717991)

[4.1. Menü 19](#_Toc179717992)

[4.1.1. Főmenü 19](#_Toc179717993)

[4.1.2. Statisztika (ha lesz adatbázis) 19](#_Toc179717994)

[4.1.3. Beállítások 19](#_Toc179717995)

[4.2. Játék 19](#_Toc179717996)

[4.2.1. Felület 19](#_Toc179717997)

[4.2.2. Interakciók (Autók és kamera mozgatása) 19](#_Toc179717998)

[5. Tesztelés, módosítások (12 oldal) 20](#_Toc179717999)

[5.1.1. Pálya generálása 20](#_Toc179718000)

[5.1.2. Járművek kezelése (lehelyezés a táblára, mozgatás) 20](#_Toc179718001)

[5.1.3. A csillag keresés 20](#_Toc179718002)

[5.1.4. Gameobject-ek létrehozása 20](#_Toc179718003)

[5.1.5. Játék ciklus 20](#_Toc179718004)

[6. Összefoglalás (1 oldal) 21](#_Toc179718005)

[Idegen nyelvű összefoglalás (1 oldal) 22](#_Toc179718006)

[Ábrajegyzék 23](#_Toc179718007)

[Irodalomjegyzék 24](#_Toc179718008)

[Melléklet 25](#_Toc179718009)

1. Bevezetés
   1. A szakdolgozat célja

Szakdolgozatom célja a Rush Hour játék elkészítése érdekes és skálázható nehézségű feladványokkal. A játék lényege, hogy az autónkat kijuttassuk egy járművekkel teli parkolóból. Habár a játékot fizikai és webes formában is megvalósították már, 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 a 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 például a visszalépéses keresés és az A csillag 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 (2-4 oldal)
   1. A játék szabályai

Az alapjáték 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 A csillag 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 az ötvenhárom 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. Adatbázis

Egy lokális sqlite adatbázissal tárolódna el az adat.

1. Szoftver tervezése (8-12 oldal)
   1. Funkciók
      1. Játék nehézségének kiválasztása

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

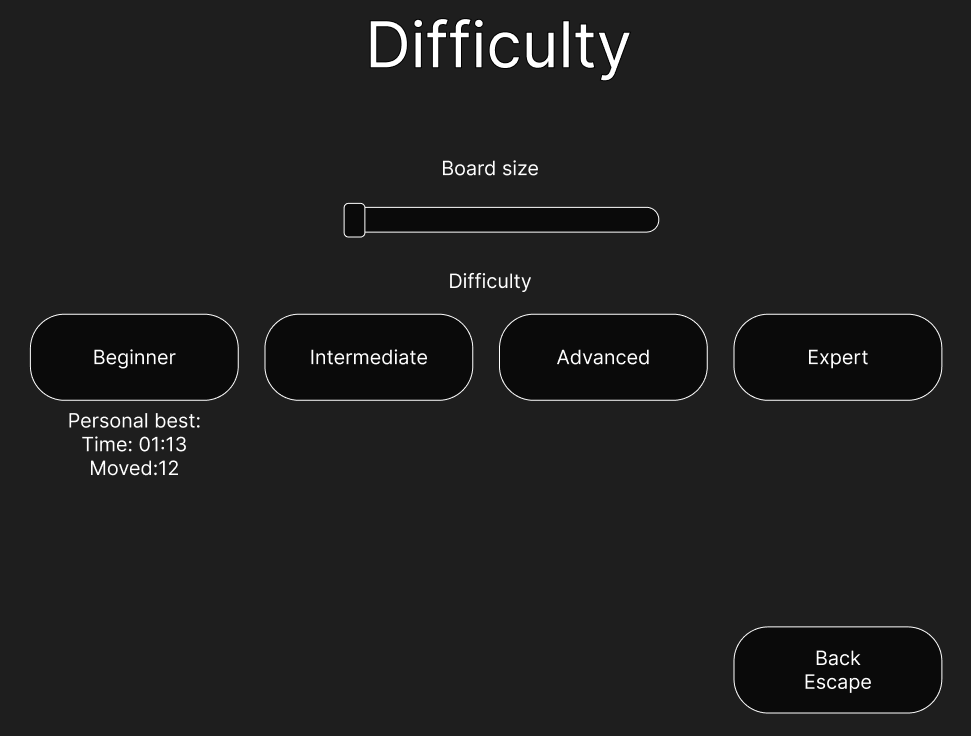
A szakdolgozatban elkészített játékban is ezeket a szinteket különböztettem meg.

A nehézség függ a pálya nagyságától és a kiválasztott szinttől. A szintek közötti különbség az autónk kezdő pozícióját módosítja úgy, hogy minél nehezebb a feladvány annál távolabb kerül a kijárattól. A generálás során a szinttől függ hány járművet helyezhetünk le a pályára és azok hogyan helyezkednek el. Minden jármű, ami a pályára kerül egy előző járműnek állja az útját akár több irányból is, a nehézség módosításával a járműveket több autó blokkolhatja.

(itt szeretnék hivatkozni egy olyan kutatásra, aminek az lett az eredménye, hogy maximum 53 lépésből bármelyik feladvány megoldható.)

A felületi elemek (1.ábra):

* Pálya mérete (slider)
* Nehézség gombok



1.ábra: Nehézség beállítások menüterv

A pálya méretét egy Slider-el lehet kiválasztani 4-től 7-ig korlátozva és a nehézségi szintet a megfelelő gomba kattintva. Az egyes nehézségek alatt megjelennek az eddig elért legjobb idő és mozgatási szám. A nehézséget kiválasztva a játék elindul a pálya generálása után.

* + 1. Játék ciklus

A játék a generálás folyamata után kezdődik és a feladvány megoldásáig vagy a felhasználó által történő megszakításig tart. A játék végén megjelenő felület a GameOver menü, amiben a feladvány megoldási ideje és lépés száma szerepel. A játékos eldöntheti, hogy folytatja-e a játékot vagy kilép a menübe.

Felületi elemek (2.ábra):

* Idő szöveg
* Lépés szám szöveg
* Új játék gomb
* Beállítások gomb
* Kilépés a menübe gomb
* Kilépés a játékból gomb

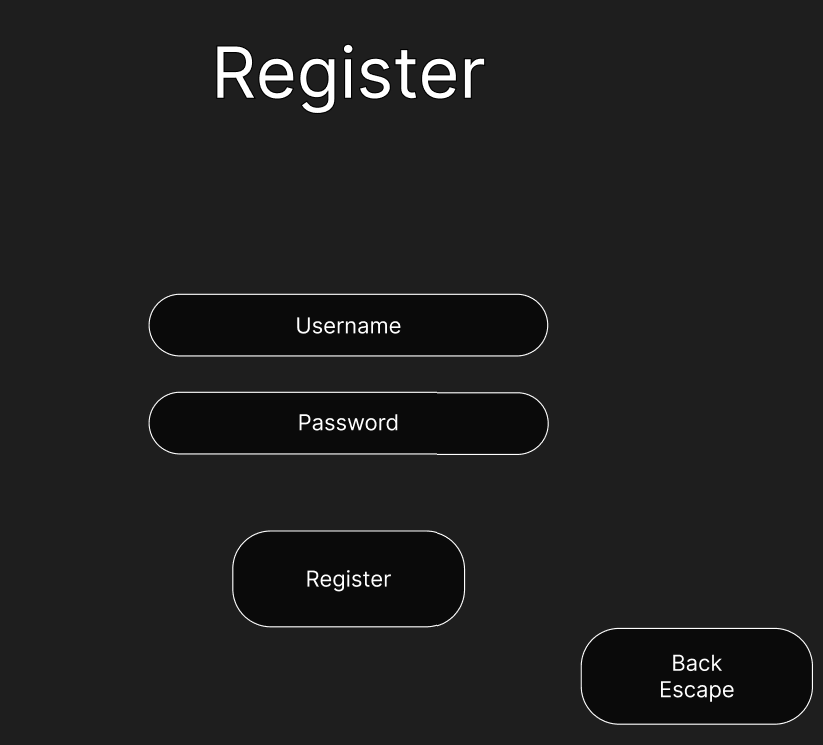


2.ábra: Játék vége felület terve

* + 1. Regisztráció

A felületi elemek (3.ábra):

* Felhasználónév beviteli mező
* Jelszó beviteli mező



3.ábra: Regisztrációs felület terve

A játék működéséhez nem szükséges regisztráció és bejelentkezés, mivel az adatbázis létrehozása során egy vendég fiókba lép be és ezt használja.

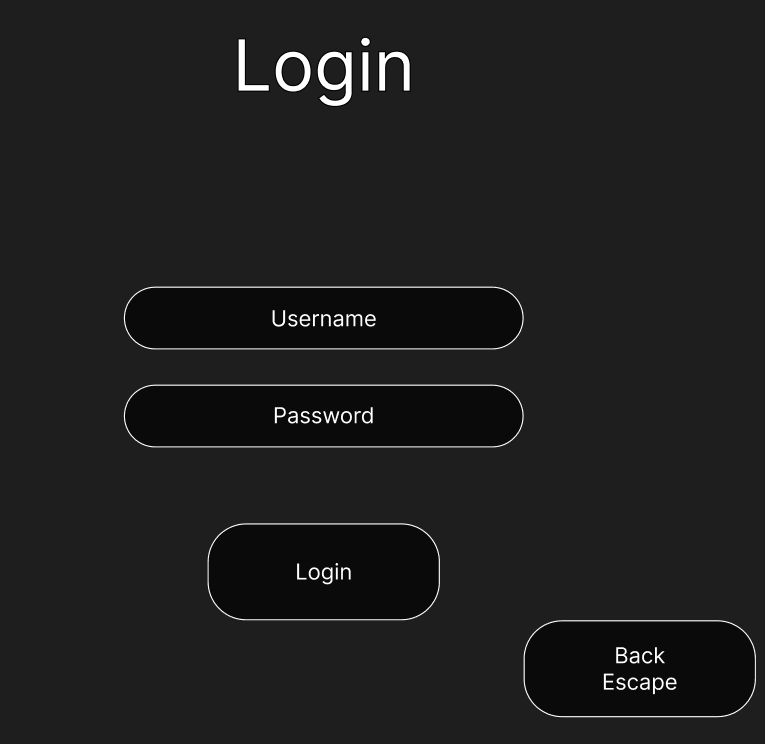
A regisztráció segít abban, ha esetleg több felhasználó szeretné a rekordjaikat vagy beállításaikat menteni, akkor ezek ne kerüljenek felülírásra a másik által és a fiókukhoz kötődjön.

A regisztráció során a felhasználó megadja a felhasználó nevét és a jelszavát, ami hash kódolva kerül az adatbázisba. A művelet sikeréről egy felugró üzenet értesíti a felhasználót. Sikertelen regisztráció történhet akkor, ha már létezik ilyen felhasználó vagy abban az esetben, ha a beírt adatok nem tesznek eleget a feltételeknek. A felhasználónévnek legalább 5 karakter hosszúnak kell lennie, a jelszónak pedig 10 karakter hosszúnak.

* + 1. Bejelentkezés

A felületi elemek (4.ábra):

* Felhasználónév beviteli mező
* Jelszó beviteli mező



4.ábra: Bejelentkezés felület terve

A játék minden indításnál a vendég felhasználóba lép be automatikusan. A bejelentkezés felületen beléphetünk egy regisztrált fiókba. A játékosnak meg kell adnia a felhasználó nevét és jelszavát. A sikeres bejelentkezés után a fő menü oldalra történik navigálás.

* + 1. Ranglista

A felületen (5.ábra) megjelennek az adatbázisba regisztrált játékosok eredményei a pálya méretétől és nehézségétől függően. A pálya méretét egy slider-el és a nehézségi szintet a lenyíló listából lehet kiválasztani. A lista változtatások után frissül és az új eredményeket mutatja. A sorrend a legkevesebb mozgatásból megoldott feladványt részesíti előnyben és utána idő szerint rendezi növekvő sorrendben.

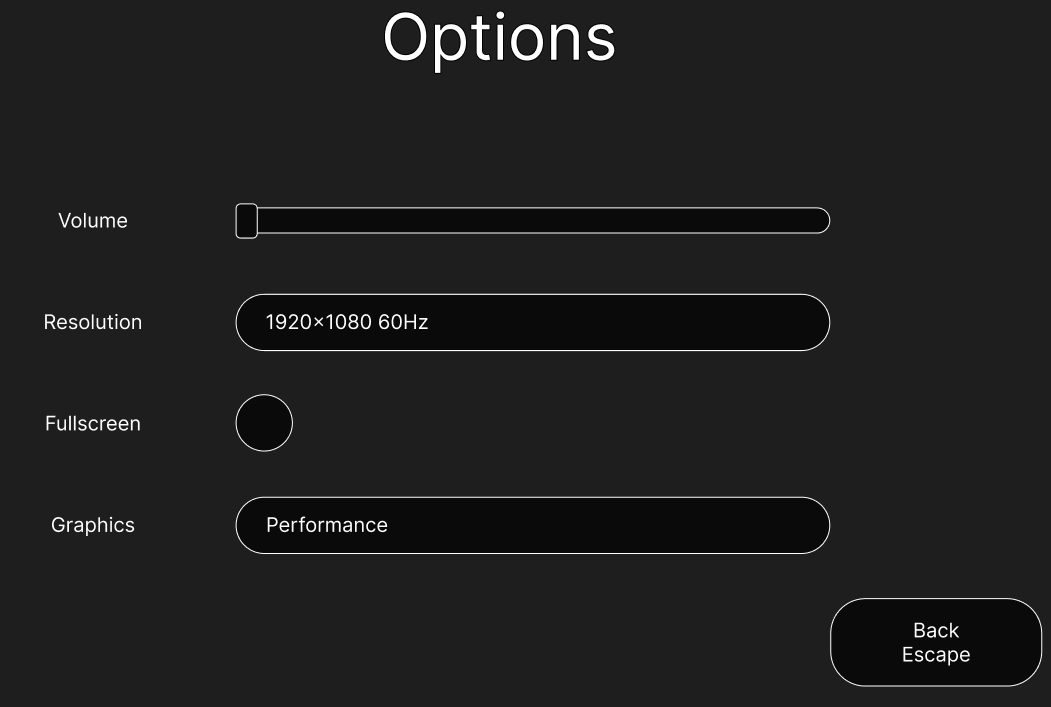


5.ábra: Ranglista felület terve

* + 1. Beállítások

Az oldalon a következő beállítási lehetőségeket lehet beállítani (6.ábra):

* Hangerő
* Felbontás
* Teljes képernyő
* Grafika



6.ábra: Beállítások felület terve

A beállításokat a bejelentkezett fiókhoz menti el, viszont ennek hiányában a vendég felhasználó beállításai íródnak felül.

A felületen a hangerő-t egy slider, a felbontást és grafikát egy lenyíló lista és a teljes képernyős módot egy checkbox segítségével lehet állítani.

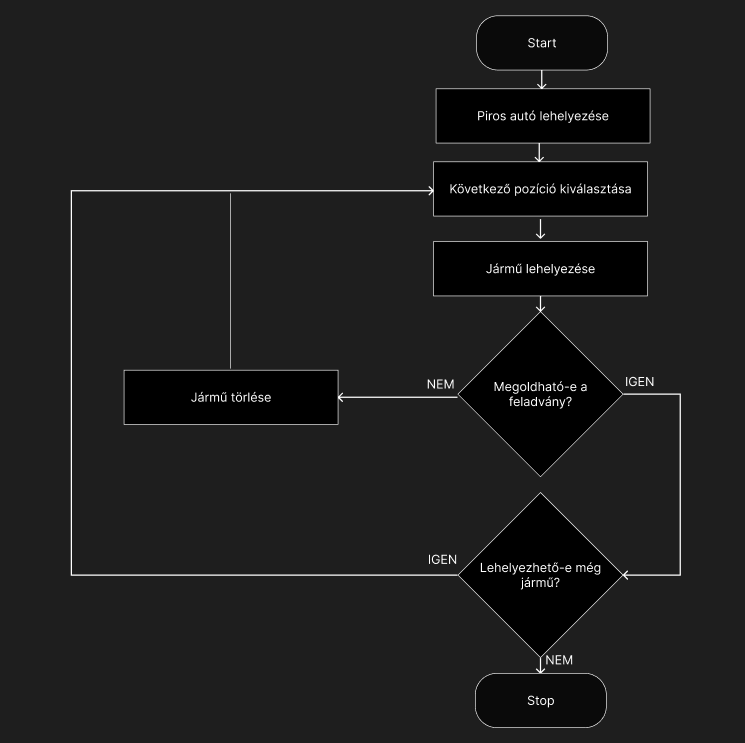
* + 1. Zene
    2. Környezeti elemek

A játék egy parkolóban helyezkedik el és ezt egy város veszi körbe, ahol autók közlekednek. Több féle autó közlekedik az utakon a közlekedési szabályokat betartva. A kereszteződésekben közlekedési lámpák irányítják a forgalmat.

A játékban napi ciklus érzékelteti az idő múlását.

* 1. Algoritmusok működése
     1. Generáló algoritmus

Az algoritmus (7.ábra) a nehézség kiválasztása után egyből elkezdődik egy töltő képernyővel jelezve a játékosnak. Az algoritmus a piros autó lehelyezésével kezdődik és a nehézség és pálya méret alapján kiválasztja a következő autó pozícióját. Ezután letett jármű blokkolja az ez előttit lehetőleg mind a két irányból. Minden jármű után lefut a megoldó algoritmus, hogy biztosan megoldható legyen a feladvány. Az a jármű törlésre kerül, amelyik miatt nem lenne megoldható a feladvány. A megoldás eldöntése után, ha már nem helyezhető le több jármű, akkor befejeződik a generálás és a töltő képernyőt is felváltja a játék felülete.



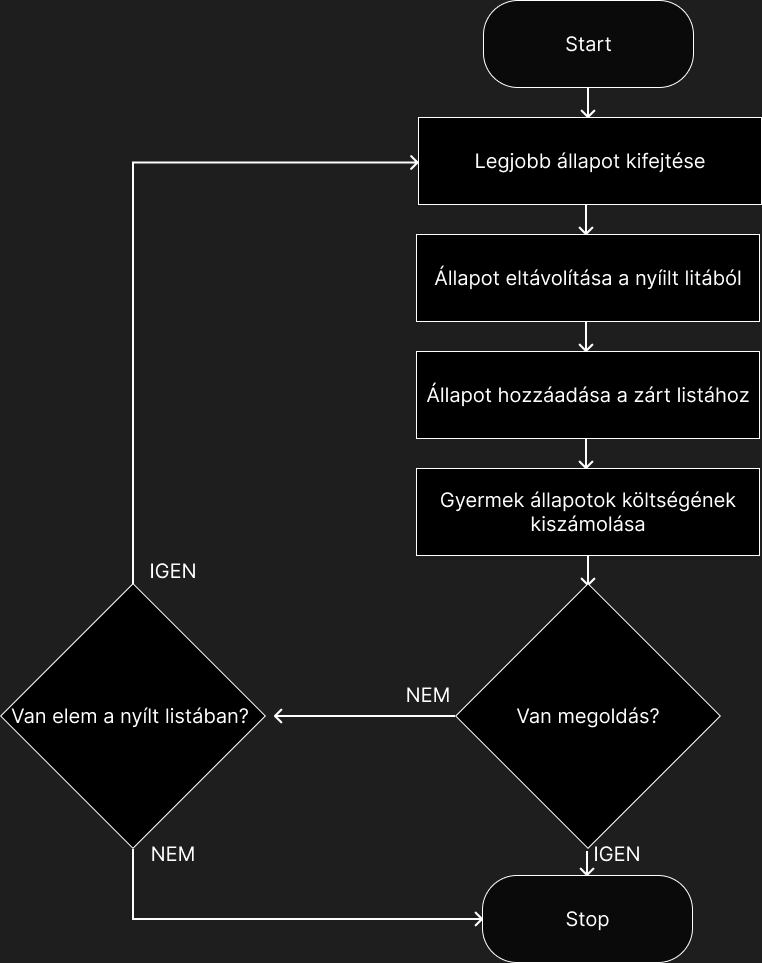
7.ábra: Generáló algoritmus folyamatábra

* + 1. Megoldó algoritmus

A solver generálás és játék közben is lefut. A generálás közben segít eldönteni, hogy egy feladvány megoldható-e és az algoritmus végén visszaadja hány lépésből oldható meg.

Játék közben a játékos a pálya jelenlegi állásából kiindulva elindíthatja a solver-t, ami megmutatja és meg is oldja a feladványt az autókat mozgatva.

Van lehetőség a feladvány visszaállítására is, ami szintén ugyanazzal az algoritmussal, visszaállítja a pályát a kezdő állapotba és nullára állítja az időt és mozgatások számát.



8.ábra: Megoldó algoritmus folyamatábra

* + 1. Járművek és pálya megjelenítése

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. Járművek és pálya megjelenítése

Miután létrejön az üres két dimenziós tömb lehelyezhető a pálya. A kijáratot külön pirossal jelöli.

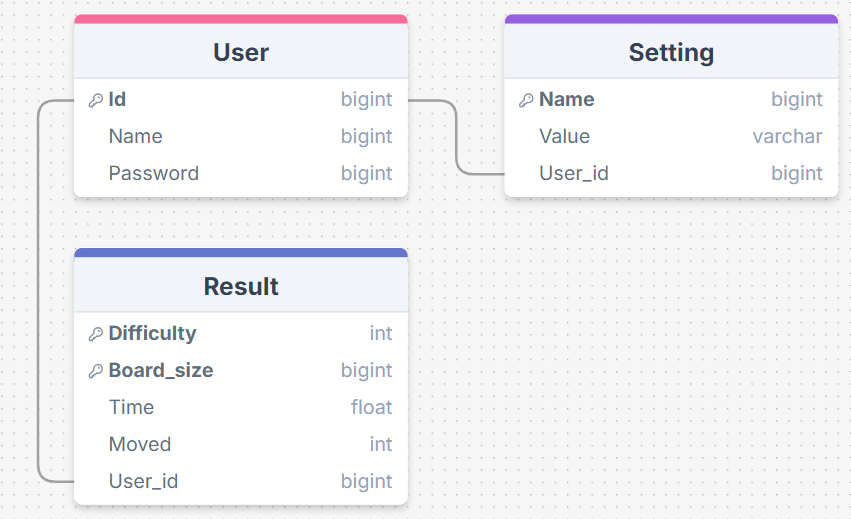
* 1. Adatbázis
     1. Adatbázis terv

SQLite adatbázisban lokálisan eltárolható adatok segítségével a játékban módosított eredmények, beállítások elmenthetők.

A játék indításakor az adatbázis törlés esetén is automatikusan létrejön és előkészíti a felhasználó számára.

Az adatbázis 3 táblából áll (9.ábra).

* felhasználó
* eredmény
* beállítás.



9.ábra: Adatbázis terv

A felhasználó táblában eltároljuk a felhasználónevet és jelszót. A jelszó hash kódolva kerül tárolásra. A játék indulásakor létrejön egy vendég felhasználó fiók, amiben minden adat eltárolódik, így nincs szükség regisztrációra a játék használatához. A funkció arra az esetre került be a játékba, ha esetleg több felhasználó szeretne egyszerre ugyanabban a játékban játszani, akkor külön elmenthetik eredményeiket.

Az eredmény táblában a pálya mérete, nehézsége és felhasználó azonosítója alapján kerül tárolásra a teljesített idő és mozgatások száma.

A beállításokban a fiókhoz kötve beállítható a játék hang ereje, felbontása, teljes képernyős módja és grafikája, amiket a játék indításakor be is tölt.

* 1. Tesztelés

Az algoritmus teszteket a Unity Test Framework segítségével lehet futtatni. A tesztek lefuthatnak Play mode-ban és Editor mode-ban.

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

Minden teszt, amire nem megvalósítható egy unit teszt megírása manuálisan lett tesztelve több tesztelő által.

1. Szoftver kialakítása (16-20 oldal)
   1. Menü
      1. Főmenü
      2. Statisztika (ha lesz adatbázis)
      3. Beállítások
   2. Játék
      1. Felület
      2. Interakciók (Autók és kamera mozgatása)
2. Tesztelés, módosítások (8-12 oldal)

☺

Ez nem végleges kép



10.ábra: Tesztelések

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

1. Összefoglalás (1 oldal)

osszefoglalaaaas

Idegen nyelvű összefoglalás (1 oldal)

szia

Ábrajegyzék

[1.ábra: nehézség beállítások menüterv 11](#_Toc179627646)

[2.ábra: játék vége felület terve 13](#_Toc179627647)

[3.ábra: Regisztrációs felület terve 14](#_Toc179627648)

[4.ábra: bejelentkezés felület terve 15](#_Toc179627649)

[5.ábra: ranglista felület terve 16](#_Toc179627650)

[6.ábra: beállítások felület terve 16](#_Toc179627651)

[10.ábra: Tesztelések 19](#_Toc179627652)

Irodalomjegyzék

Melléklet