Békéscsabai SZC Nemes Tihamér Technikum és Kollégium

Szakképesítés megnevezése: Szoftverfejlesztő

Azonosító száma: 54 213 05

**Záródolgozat**

Off time

Konzulens Neve Készítette: Varga András Bendegúz

Gajda Judit 5/13.B

Oktatási azonosító: 72169956207

Békéscsaba, 2021/2022

Tartalom

[1. Bevezetés 5](#_Toc100834110)

[1.1 Köszönetnyilvánítás 5](#_Toc100834111)

[1.2 Indoklás 5](#_Toc100834112)

[2. Fejlesztői dokumentáció 5](#_Toc100834113)

[2.1 Fejlesztői környezet 5](#_Toc100834114)

[2.1.1 Programozási nyelv és környezet 5](#_Toc100834115)

[2.1.2 Játékmotor 6](#_Toc100834116)

[2.2 Adattárolás 6](#_Toc100834117)

[2.2.1 Adatbázisszerkezet 6](#_Toc100834118)

[2.2.2 A users tábla 6](#_Toc100834119)

[2.2.3 A saves tábla 6](#_Toc100834120)

[2.2.4 A wins tábla 7](#_Toc100834121)

[2.2.5 A questions tábla 7](#_Toc100834122)

[2.2.6 Fájlrendszer 7](#_Toc100834123)

[2.2.7 Mentés menete 7](#_Toc100834124)

[2.2.8 Betöltés menete 8](#_Toc100834125)

[2.3 Adatbázis kezelés 9](#_Toc100834126)

[2.3.1 Regisztráció és bejelentkezés 9](#_Toc100834127)

[2.3.2 Mentés 9](#_Toc100834128)

[2.3.3 Betöltés 10](#_Toc100834129)

[2.3.4 Szintteljesítés 10](#_Toc100834130)

[2.3.5 Ranglista 10](#_Toc100834131)

[2.3.6 Kvíz kérése 11](#_Toc100834132)

[2.3.7 Kvízek megtekintése, módosítása 11](#_Toc100834133)

[2.4 Játékelemek 11](#_Toc100834134)

[2.4.1 Játékos drónja 11](#_Toc100834135)

[2.4.2 Energiagócok 12](#_Toc100834136)

[2.4.2.1 Forgatható gócok 12](#_Toc100834137)

[2.4.2.2 Logikai gócok 12](#_Toc100834138)

[2.4.2.3 Blokkoló gócok 12](#_Toc100834139)

[2.4.2.4 Cél góc 13](#_Toc100834140)

[2.4.2.5 Ajtónyitó 13](#_Toc100834141)

[2.4.2.6 Ajtó 13](#_Toc100834142)

[2.4.2.7 Csatlakozó 13](#_Toc100834143)

[2.4.3 Kvíz ablak 13](#_Toc100834144)

[2.4.4 Szünet ablak 14](#_Toc100834145)

[2.5 Tesztelés 14](#_Toc100834146)

[2.5.1 Tesztelés módja 14](#_Toc100834147)

[2.5.2 Talált hibák 14](#_Toc100834148)

[2.5.3 Egyéb megjegyzések, visszajelzések 15](#_Toc100834149)

[2.5.4 Tesztelési platform 15](#_Toc100834150)

[2.6 Adatbiztonság 16](#_Toc100834151)

[3. Felhasználói dokumentáció 16](#_Toc100834152)

[3.1 Minimum rendszerkövetelmények 16](#_Toc100834153)

[3.2 Telepítés 16](#_Toc100834154)

[3.3 A játékról 17](#_Toc100834155)

[3.3.1 Cím 17](#_Toc100834156)

[3.3.2 Sztori 17](#_Toc100834157)

[3.4 Játék indítása 17](#_Toc100834158)

[3.4.1 Új felhasználó létrehozása 17](#_Toc100834159)

[3.4.2 Bejelentkezés 18](#_Toc100834160)

[3.4.3 Új játék indítása 19](#_Toc100834161)

[3.5 Játékmenet 19](#_Toc100834162)

[3.5.1 Irányítás 19](#_Toc100834163)

[3.5.2 Játék célja 20](#_Toc100834164)

[3.5.3 Forgatható gócok 20](#_Toc100834165)

[3.5.4 Nem forgatható gócok 20](#_Toc100834166)

[3.5.5 Logikai gócok 21](#_Toc100834167)

[3.5.6 Blokkoló gócok 21](#_Toc100834168)

[3.5.7 Ajtónyitó gócok 22](#_Toc100834169)

[3.5.8 Szint teljesítése 22](#_Toc100834170)

[3.5.9 Szünet menü 23](#_Toc100834171)

[3.6 Egyéb funkciók 24](#_Toc100834172)

[3.6.1 Mentés 24](#_Toc100834173)

[3.6.2 Betöltés 26](#_Toc100834174)

[3.6.3 Kérdések módosítása és hozzáadása 27](#_Toc100834175)

[3.6.4 Ranglista 28](#_Toc100834176)

[4. Összefoglalás 29](#_Toc100834177)

[4.1 Szakdolgozat célja 29](#_Toc100834178)

[4.2 Megvalósítás 29](#_Toc100834179)

[4.2.1 Puzzle rendszer 29](#_Toc100834180)

[4.2.2 Karakter kontroller alapjai 29](#_Toc100834181)

[4.2.3 Kvíz rendszer 30](#_Toc100834182)

[4.2.4 Mentés és betöltés alapjai 30](#_Toc100834183)

[4.2.5 Felhasználókezelés 30](#_Toc100834184)

[4.2.6 Dinamikus lista menü 30](#_Toc100834185)

[4.2.7 Mentés és betöltés a dinamikus lista menüből 31](#_Toc100834186)

[4.2.8 Megvilágítás 31](#_Toc100834187)

[4.2.9 Hologram shader 31](#_Toc100834188)

[4.2.10 Dinamikus lista rendezés 31](#_Toc100834189)

[4.2.11 Kód dokumentáció 32](#_Toc100834190)

[4.2.12 Adatbázis feltöltése kérdésekkel 32](#_Toc100834191)

[4.2.13 Pályák elkészítése 32](#_Toc100834192)

[4.3 Fejlesztési lehetőségek 33](#_Toc100834193)

[4.3.1 Kinézet fejlesztése 33](#_Toc100834194)

[4.3.2 Hangok hozzáadása 33](#_Toc100834195)

[4.3.3 Több pálya és kérdés hozzáadása 33](#_Toc100834196)

[4.3.4 Adaptálás mobil eszközökre 33](#_Toc100834197)

[4.3.5 Online belépés 34](#_Toc100834198)

[4.3.6 Online ranglista 34](#_Toc100834199)

[4.3.7 Kvízek online megosztása 34](#_Toc100834200)

[4.3.8 Pályaszerkesztő hozzáadása 34](#_Toc100834201)

[4.3.9 Nyelvi lokalizáció 34](#_Toc100834202)

[5. Irodalomjegyzék 35](#_Toc100834203)

# 1. Bevezetés

## 1.1 Köszönetnyilvánítás

Ezúton szeretnék köszönetet nyilvánítani tanáraimnak, akik itt létem alatt előkészítették tudásomat, és a programozás iránti érdeklődést felkeltették bennem.

## 1.2 Indoklás

Azért választottam ezt a témát, mert a programozás mindig is közelebb állt hozzám, mint a webfejlesztés, és már régóta szerettem volna saját játékprogramot készíteni. Már a záródolgozat feljövetele előtt célom volt az, hogy C# nyelven megtanuljak programozni, de ebben a legtöbb esetben limitált és unalmas online talált feladatok nem sokat segítettek. Ezzel ellenben, egy logikai fejtörő/kvízjáték elkészítése megfelelő kihívásnak tűnt a képességeim fejlesztéséhez. Sok hasonló fejlesztő és ismeretterjesztő tartalom létezik már, de a választék bővítése sosem árt, és egy interaktív játékprogram könnyebben megtartja az emberek figyelmét, mint például egy weboldal.

# 2. Fejlesztői dokumentáció

## 2.1 Fejlesztői környezet

### 2.1.1 Programozási nyelv és környezet

A programot C# nyelven írtam, hiszen a Unity ezt használja. A nyelv alapból is jó választás lett volna, mivel széleskörűen támogatott, objektum orientált és sok hasznos alapkönyvtárral rendelkezik. Mindezek mellett egy könnyű nyelv, nem olyan gépközeli, mint például a C++, a program futásával kapcsolatos dolgokat kezeli helyettünk, de futási sebessége mégis elég gyors. Fejlesztési környezetnek Visual Studio-t használtam, ami a Unity-vel összekapcsolódva még könnyebbé teszi e fejlesztést.

### 2.1.2 Játékmotor

Játékmotornak a Unity-t választottam, mivel 2D játékok készítéséhez kitűnő, és a scriptekhez C# nyelvet használ, amiben már van egy kis jártasságom. A program könnyen kezelhető, és alapból kezeli a legtöbb szükséges szimulációt, nekem csak a játékelemekkel és azok interakciójának logikájával kellett foglalkoznom.

## 2.2 Adattárolás

### 2.2.1 Adatbázisszerkezet

A felhasználói adatok és kvíz kérdések tárolására a program a GameDB nevű adatbázist használja. Minden adatbázissal kapcsolatos operációért a DatabaseManager osztály felel. Minden osztály, aminek szüksége van az adatbázisra ezen keresztül szerzi meg a szükséges adatokat.

### 2.2.2 A users tábla

Ez a tábla a felhasználók tárolásáért felelős. A felhasználók nevein és hashelt jelszavain kívül még azt is tárolja, hogy hányadik pályánál tartanak.

### 2.2.3 A saves tábla

A saves tábla a mentéseket tartja számon. A userID oszlop tárolja, hogy melyik felhasználóhoz tartozik a mentés. A levelName a pályanevet tárolja, a fileName pedig a mentésfájl kiterjesztés nélküli nevét. Ezek mellett tárolja a nehézséget, mozgások számát, eltelt időt és a mentés idejét.

### 2.2.4 A wins tábla

A felhasználó pályateljesítéseinek adatait tárolja. Ugyanúgy, mint a saves táblánál, a felhasználóra a userID idegen kulccsal hivatkozunk. Tárolja a nehézséget, lépések számát, pályanevet, eltelt időt és a teljesítés idejét is.

### 2.2.5 A questions tábla

Ez a tábla tárolja a kvízeket. Egy kvíz rekord áll egy azonosítóból, nehézségből, kvíz szövegből, jó válaszból és három rossz válaszból. Ezek mellett azt is tárolja, hogy a rekordot felhasználó hozta-e létre.

### 2.2.6 Fájlrendszer

A program az adatbázist és a mentés mappát a felhasználó AppData/LocalLow/Varga András Záródolgozat/ mappán belül tárolja. A felhasználók mentéseit a mentés mappában tárolja, és ezeket az adatbázis saves táblájának segítségével kezeli. Ha az adatbázist kitörölték, vagy még nincsen létrehozva, akkor indításkor azt létrehozza, és az alap kvízekkel fel is tölti.

### 2.2.7 Mentés menete

A mentés folyamata akkor kezdődik, amikor a felhasználói felületről egy gombnyomással meghívjuk a GameController objektum Save metódusát. Először címkéjük alapján megkeressük az összes menteni való objektumot, majd egy listába eltároljuk az összes objektum ISaveable interfészének referenciáit.



Ezután példányosítunk egy új SaveData objektumot, és az összes menteni való objektummal hozzáadatjuk a saját adatát.



Az így elkészült SaveData objektumot ezután szerializáljuk, és a FileManager objektum segítségével fájlba írjuk.

### 2.2.8 Betöltés menete

Betöltéskor a kiválasztott mentés adatai bekerülnek a PlaySession-be, és az abban szereplő szint betöltésre kerül. A szint betöltésekor a GameController látja, hogy a PlaySession tartalmaz betöltendő fájlt, és a FileManager segítségével beolvassa azt. A fájlt ezután deszerializálja és egy SaveData objektumként eltárolja. Ezután az összes menthető objektumon meghívjuk az ISaveable interfész LoadFromSave metódusát ezzel az új SaveData objektummal paraméterként, és visszatöltetjük vele a mentett állapotát.

## 2.3 Adatbázis kezelés

Az adatbázissal kapcsolatos operációkért a DatabaseManager osztály felelős.

### 2.3.1 Regisztráció és bejelentkezés

Mindkét operáció felhasználói felületi részét a MainMenuManager objektum kezeli. Ellenőrzi, hogy minden mező ki van-e töltve, nem tartalmaz speciális karaktert, és a hosszuk is megfelelő, és csak ezután hívja meg a beírt paraméterekkel a DatabaseManager megfelelő metódusait.

Regisztrációnál a RegisterUser metódus le ellenőrzi, hogy a megadott felhasználónévvel létezik-e már felhasználó, és ha nem, akkor beregisztrálja azt. Ezután a metódus visszatérési értéke szerint a MainMenuManager visszajelzést ad a felhasználónak.

A bejelentkezés is hasonló módon működik, de itt elég lekérdezni, hogy a megadott felhasználónév és jelszópáros létezik-e, és ha igen, akkor a felhasználó azonosítóját és jelenlegi szintjét eltároljuk a statikus PlaySession osztályban.

Kijelentkezéskor csak visszaléptetjük a felhasználót a bejelentkező képernyőre és kiürítjük a PlaySession-t.

### 2.3.2 Mentés

A mentéshez tartozó felhasználói felületet a DynamicListManager objektum kezeli. A mentés menü megnyitásakor a DatabaseManager GetSavedGames metódusával lekéri a felhasználó mentés rekordjait, majd ezeket kiírja.

Új mentés készítésekor a DynamicListManager le ellenőrzi, hogy a megadott mentésnév jó-e, majd a DatabaseManager CheckIfSaveNameTaken metódusával megnézi, hogy foglalt-e. Ha a név jó, akkor a GameController objektum Save metódusának meghívásával megpróbálja fájlba mentetni a pillanatnyi játékállást. Ha a mentés sikeres, akkor a mentésnév és a mentésfájl neve a PlaySession-be bekerül, és annak adataival a DatabaseManager AddSave metódusával új mentés rekordot vesz fel.

A mentések felülírása is hasonlóan működik, csak új mentésfájl és rekord létrehozása helyett a kiválasztott mentésfájlt a FileManager felülírja, majd a mentés rekordot a DatabaseManager OverwriteSave metódusa felülírja.

Mentések törlésekor a FileManager DeleteFile metódusával kitöröljük a mentésfájlt, majd a DatabaseManager DeleteSave metódusával kitörökjük a rekordot is. Ha a kitörölt mentés ugyanaz volt, amiről betöltöttük a játékot, akkor a PlaySession saveTitle és fileName attribútumait kitöröljük.

### 2.3.3 Betöltés

A betöltés folyamata a DynamicListManager objektumban kezdődik. A mentett játékokat a DatabaseManager GetSavedGames metódusával kérdezzük le. Egy mentés kiválasztásakor annak adatai bekerülnek a PlaySession osztályba, majd az ebben lévő szint a UIEventHandler SwitchScene metódusának meghívásával betöltésre kerül.

### 2.3.4 Szintteljesítés

A szintteljesítés folyamata a GoalScript objektumokban indul. Hogyha a teljesített szint száma megegyezik a játékos PlaySession currentLevel értékével, akkor azt eggyel meg kell növelni, és azt a DatabaseManager UpdateUserCurrentLevel metódusával az adatbázisban is frissítjük. A felhasználó új eredménye az AddSave metódussal kerül felvételre.

### 2.3.5 Ranglista

A ranglista felületét a DynamicListManager kezeli. A DatabaseManager GetLeaderBoard metódusával lekéri az összes felhasználó eredményeit.

### 2.3.6 Kvíz kérése

A játékon belül a kvízek kezeléséért a QuizHandler osztály felel. Amikor a felhasználónak meg kell válaszolnia egy kvízt, a QuizHandler a DatabaseManager GetQuestion metódusával kér le egy kvízt.

### 2.3.7 Kvízek megtekintése, módosítása

Ezekért a funkciókért is a DynamicListManager felel. A rekordokat a DatabaseManager GetUserQuizes metódussal kéri le, majd ezeket kilistázza. Rekordok törlésére a DeleteQuiz metódus szolgál.

A rekordok bővítésekor vagy módosításakor a RecordAdder osztályt használjuk. Ennek az osztálynak is külön felülete van, amit kezelhet. A felület a megnyitásakor a DynamicListManager meghívja a RecordAdder Refresh metódusát. Ez egy túlterhelt metódus, így amikor új kvízt akarunk megadni, akkor paraméterek nélkül meghívva kiüríthetjük vele a beviteli mezőket, de ha már meglévő kvízt akarunk módosítani, akkor paraméterként az azt reprezentáló QuizData megadásával beírja annak adatait a beviteli mezőkbe, hogy azokat csak javítani, módosítani kelljen újra begépelés helyett.

## 2.4 Játékelemek

### 2.4.1 Játékos drónja

A drón amit a játékos irányít, a DroneController osztályt használja. A WASD gombokat használja az irányításra. Ezeknek a gomboknak az állapotából készített Vector2 irányába állítjuk a Rigidbody2D lendületét, és ugyanebbe az irányba forgatjuk a Tansform-ját. A játékon belüli kattintásokat is ez kezeli. Kattintáskor a kamerából indulva egy Raycast2D-t indítunk az egér irányába, és ha eltalálunk valamit, ami implementálja az IConnectable interfészt, akkor annak meghívjuk a Click metódusát a drónt megadva paraméterként.

### 2.4.2 Energiagócok

A játék puzzle része különböző típusú gócokon alapul, amiket PowerConnection osztályt használó játékelemek kapcsolnak össze. A cél az, hogy a háromszög alakú forrásból kiindulva áramot juttassunk a kör alakú célba.

### 2.4.2.1 Forgatható gócok

Vannak I, L, és T alakú gócok, amiket a játékos kattintással forgathat, kivéve, ha nem forgathatók (sötét zöldes szín), vagy egy blokkoló miatt le vannak zárva (piros szín). A forgatásért a Turn metódus felelős. A lezárt gócokat az azokhoz csatlakozó blokkoló feloldásával tehetjük elérhetővé. A forgatható és logikai gócok a PowerNode osztályt használják.

### 2.4.2.2 Logikai gócok

Fajták: NOT, AND, OR, NAND, NOR, XOR, XNOR. Ezek a gócok akkor lesznek aktívak, amikor a bemeneteik állapotán végzett logikai operáció igaz.

### 2.4.2.3 Blokkoló gócok

Feloldásukhoz először áramot kell vezetni beléjük, majd megfelelő távolságból rákattintás után meg kell válaszolni egy kvízt. Ha túl messze vagyunk, akkor ez a góc is piros. A kvíz sikeres megválaszolása után a blokkoló és az általa blokkolt góc feloldódik. A blokkolók a NodeBlocker osztályt használják.

### 2.4.2.4 Cél góc

A cél góc árammal való ellátásával teljesíthető a szint. Ilyenkor megáll a játék és megjelennek a felhasználó eredményei, feloldódik a következő pálya. A GoalScript osztályt használja.

### 2.4.2.5 Ajtónyitó

A mellette lévő ajtót irányítja. Ha áramot kap kinyitja, ha nem kap áramot becsukja. A DoorController osztályt használja.

### 2.4.2.6 Ajtó

A DoorController által irányított Door osztályt használja. Nyitáskor és csukáskor az ajtó child GameObject-jének enbled attribútumát állítja annak SetActive metódusával.

### 2.4.2.7 Csatlakozó

A PowerConnecion osztály a gócok és más játékelemek csatlakoztatásáért felelős. Amikor a bemenetének megváltozik az állapota, színt vált, és a kimenetét frissíti és közli vele az új állapotát.

### 2.4.3 Kvíz ablak

A blokkolók feloldásához meg kell válaszolni egy kvízt, amit a QuizHandler kezel. Blokkolóra kattintáskor a QuizHandler kér egy QuizData-t az adatbázisból, kisorsolja a válaszok elrendezését, majd behelyettesíti az összes szövegdoboz szövegét. Ekkor megáll a játékon belüli idő (Az időzítő még mindig számol), és megnyílik a kvíz felület. Ezután a felhasználónak az egyik gombra kattintva válaszolnia kell. Ha a jó válasz gombjára kattint, akkor a gomb háttere kék lesz, és a blokkoló feloldódik. Az ellenkező esetben a gomb háttere piros lesz, és a blokkoló zárva marad. A felületet a felhasználó a vissza gombbal vagy az Esc billentyű megnyomásával bármikor bezárhatja.

### 2.4.4 Szünet ablak

A játékos az Esc billentyű megnyomásával bármikor felfüggesztheti a játékot, és megnyithatja a szünet menüt. Ebből a menüből meg lehet nyitni a mentés és betöltés menüket, vissza lehet lépni a főmenübe, és be is lehet zárni a programot. A játék az Esc billentyű megnyomásával folytatható.

## 2.5 Tesztelés

### 2.5.1 Tesztelés módja

A teszteléshez Discordon keresztül a barátaimtól, ismerőseimtől és egyéb emberektől kértem segítséget. A szerverünkön van egy „@Tester” rang, amit bárki kérhet, és külön szoba, ahova a tesztelni való dolgainkat beposztolhatjuk, ilyenkor az összes Tester értesítést kap. A programot tömörített fájlba becsomagoltam, feltöltöttem a Google drive-omra, és a fájl linkjét beposztoltam az előbb megemlített szobába. Ezt a fájlt többen is letöltötték és kipróbálták.

### 2.5.2 Talált hibák

A tesztelés csak egy működésbeli hibát tárt fel. Ha a játékos a drónnal eltakarja azt a gócot vagy blokkolót, amire rá szeretne kattintani, akkor nem történik semmi. Ez a probléma könnyen orvosolható volt azzal, hogy a kattintás Raycast2D-jének adtam egy LayerMask-ot, ami csak a gócokra és blokkolókra tesztel.

Volt még egy pár helyesírási hiba a kérdésekben, de azok inkább az excel helyesírásellenőrzőjének hibái voltak, például nem tudom, hogy a „módisító” mióta érvényes magyar szó. Ezeket a hibákat az adatbázison belül gyorsan kijavítottam.

Többször is előfordult az, hogy egymás után kétszer ugyanazt a kvízt kapta a felhasználó. Ennek annyit lehetne csinálni, hogy az adatbázisba több kérdést teszünk. Minden nehézséghez van 20-20 kérdés, így aránylag nagy esély van az újra sorsolásra. A játék rövid hossza és a kérdések kitalálásának időigényessége miatt úgy döntöttem, hogy az extra kérdések hozzáadását a felhasználókra bízom.

### 2.5.3 Egyéb megjegyzések, visszajelzések

Többen is megjegyezték, hogy a mozgás reszponzív volt, és a drón forgása gyors volt, de egyenletes. Voltak, akik több percen át azzal szórakoztak, hogy az egyhelyben forgással kísérleteztek.

Az is tetszett nekik, hogy a gócok és az azokat összekötő csatlakozások mindig láthatók. Ha ezek is eltűnnének a sötétben, akkor sokkal nehezebb lenne átlátni a pályát, és ez csak több nehézséghez és frusztrációhoz vezetne, nem adna semmit a játékélményhez.

Egy tesztelő azt mondta, hogy szerinte jobb, hogy a blokkolókat nem kell áram alatt tartani ahhoz, hogy az általuk blokkolt góc nyitva maradjon. Ez is csak egy nem kívánt komplexitás lett volna.

Egy másik tesztelő az árnyékok minőségéről tett megjegyzést. A játék egyszerű grafikáját látva nem számított arra, hogy hasonló dolgok bármi figyelmet kaptak volna. Szerinte ez a kis megvilágítás igazán nem nagy dolog, de jobb, mintha a pálya teljes megvilágításban vagy sötétségben lenne. Ezek mellett emiatt eggyel több dolog is történik a képernyőn, az alapból nem túl összetett látványt egy kicsit érdekesebbé teszi.

### 2.5.4 Tesztelési platform

Az összes tesztelő 64 bites Windows 10 alatt futtatta a programot, ami azonnal működött, nem kellet semmi extra programot vagy keretrendszert telepíteniük.

## 2.6 Adatbiztonság

A felhasználónak egy felhasználónéven és jelszón kívül nem kell semmi adatot megadnia, és minden adat lokálisan van tárolva. A jelszavak Hash-elve vannak, és az adatbázisban akkor sem egyeznek, ha két felhasználó ugyan azt a jelszót adta meg.

Mivel a Unity ingyenes verzióját használom, a program az első induláskor hardver adatokat gyűjt, és ezeket névtelenül vissza akarná küldeni. Erre azért van szükség, hogy statisztikákat készítsenek arról, hogy a Unity felhasználóinak milyen erős számítógépei vannak, hogy a motort jobban az igényekhez szabják. Ez a funkció viszont tűzfal hozzáférést igényel, így amikor hozzáférést kér, a „Mégsem” gombra kattintással letiltható.

# 3. Felhasználói dokumentáció

## 3.1 Minimum rendszerkövetelmények

* CPU: Bármilyen Intel Core i3 processzor
* RAM: 200MB
* Tárhely: 100MB
* VGA: Integrált videó kártya
* Operációs rendszer: Windows 7, 10, 11

## 3.2 Telepítés

A telepítő megnyitása után megnyílik a telepítő varázsló. A telepítési hely megadása és a licenc elfogadása után a program a kijelölt helyre telepítésre kerül, és készen lesz az indításra. Indításkor a játék tűzfalhozzáférést kér, ezt el lehet utasítani.

## 3.3 A játékról

### 3.3.1 Cím

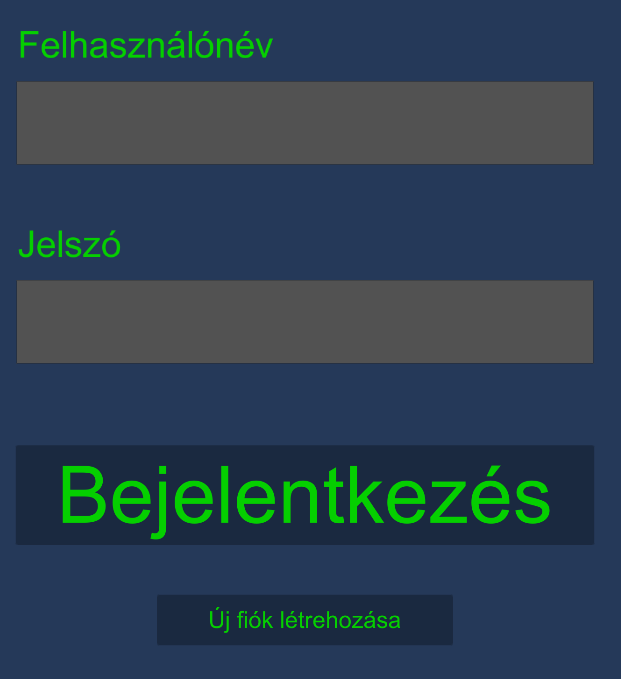
Az angol „off time” kifejezés jelenthet nem munkával töltött időt, vagy azt az időintervallumot, amikor egy elektromos eszköz nincsen bekapcsolva.

### 3.3.2 Sztori

Ön egy rendszeradminisztrátor, aki a vakációja kellős közepén hívást kap a főnökétől, amiből kiderül, hogy az egész cégterületen elment az áram. A cég távoli menedzsment szoftverébe belépve kiderül, hogy néhány saját akkumulátorral rendelkező karbantartó drónon kívül semmi nem elérhető, így azokat használva kell elhárítani a hibát.

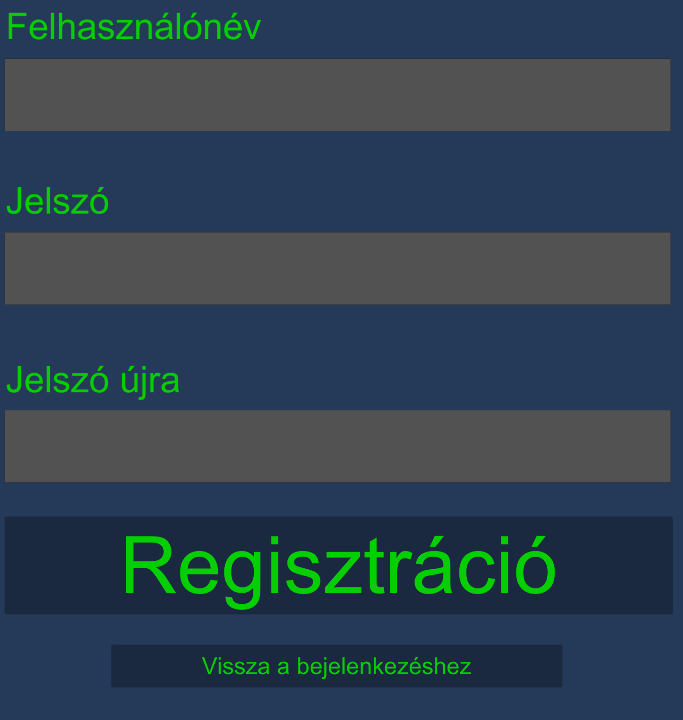
## 3.4 Játék indítása

### 3.4.1 Új felhasználó létrehozása



Indításkor a felhasználót a bejelentkezés ablak várja. Ahhoz, hogy belekezdhessünk a játékba, előbb be kell jelentkeznünk egy fiókba. Mivel alapból nincsen fiókunk, regisztrálnunk kell egyet.

A regisztrációs felület megjelenítéséhez az „Új fiók létrehozása” gombra kell kattintani

A regisztráláshoz meg kell adni egy felhasználónevet és egy jelszót. Ezután a „Regisztráció” gombra kattintva regisztrálhatunk.

Ha egy mező rosszul van kitöltve, vagy kitöltetlen marad, akkor a kérdéses mező alatt egy piros téglalapban megjelenő szöveg megmondja, hogy mit csináltunk rosszul.

Ha minden jól megy, akkor a gomb alatt zöld téglalapban a „Sikeres regisztráció” szöveg jelenik meg.

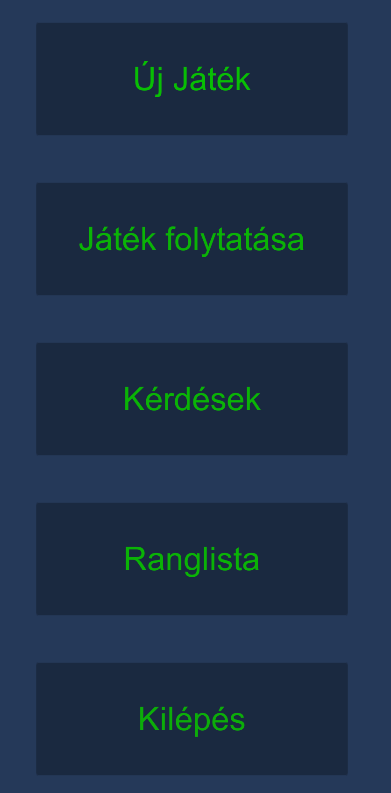
A „Vissza a bejelentkezéshez” gombra kattintva léphetünk vissza a bejelentkező ablakhoz.

### 3.4.2 Bejelentkezés

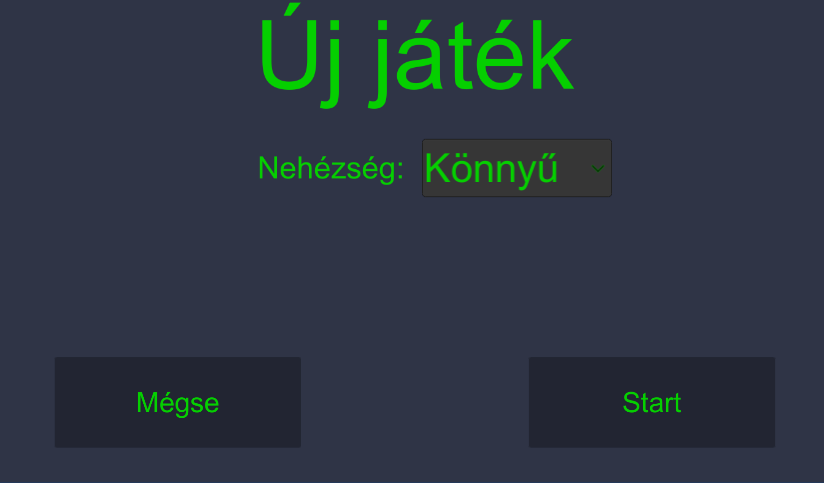
Bejelentkezéshez csak meg kell adni az előbb regisztrált adatokat, és meg kell nyomni a „Bejelentkezés” gombot. Ha valamit rosszul adtunk meg, akkor a regisztrációs felülethez hasonló módon visszajelzést kapunk.

Helyesen megadott adatok esetén megtörténik a bejelentkezés, és megnyílik a főmenü.

### 3.4.3 Új játék indítása

 A főmenübe lépve elénk tárul a játék összes funkciója. Új játék indításához az „Új játék” menüpontra kell kattintani. Ekkor megnyílik a szintválasztó felület, ahol még csak az első szint lesz elérhető.

A szint kiválasztása után megjelenik egy párbeszédablak, amin ki kell választanunk a kívánt nehézséget. Ha ezt megtettük, akkor a „Start” gomb megnyomásával kezdhetjük meg a játékot.



## 3.5 Játékmenet

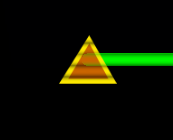
### 3.5.1 Irányítás

Mozgás: WASD

Kattintás az egérrel

### 3.5.2 Játék célja

A forrás gócból a cél gócba kell eljuttatni az áramot.

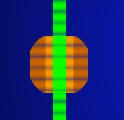
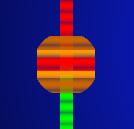


Cél góc

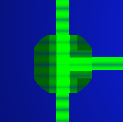
Forrás góc

### 3.5.3 Forgatható gócok

Azáram folyását az elforgatható sárga gócok forgatásával lehet befolyásolni. Ezek lehetnek I, L és T alakúak.

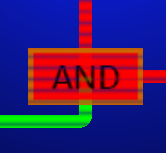


### 3.5.4 Nem forgatható gócok



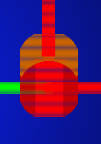
Az előbb említett gócok közül nem mind forgatható. Ezeket a sötétebb zöldes színükről lehet felismerni.

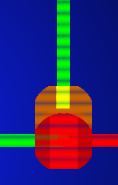
### 3.5.5 Logikai gócok



A logikai gócok NOT, AND, OR, NAND, NOR, XOR, és XNOR változatokban fordulnak elő. A NOT kapu kivételével az összes hasonló gócnak két bemenete van. Ha az ezeken elvégzett logikai operáció igaz, akkor a góc aktív lesz.

### 3.5.6 Blokkoló gócok

 Vannak olyan gócok, amik blokkolva vannak. Ezeket arról lehet felismerni, hogy pirosok, és hozzájuk van csatolva egy blokkoló. A blokkoló feloldásakor a blokkolt góc is feloldódik. A forgatható gócokkal ellentétben a blokkolókat nem lehet bárhonnan aktiválni, mivel előre megszabott hatótávolságuk van. Ha ön túl messze van, akkor a blokkoló is bepirosodik.

Blokkolt gócok feloldásának menete:

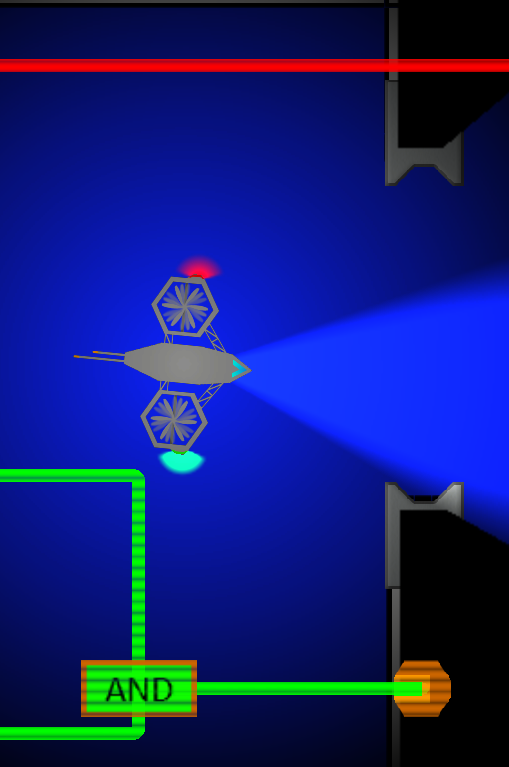
1, A blokkolóba áramot vezet. Ilyenkor a csatlakozó sárga színre vált.

2, Közel megy a blokkolóhoz, ha túl messze van.

3, A blokkolóra kattint. Ekkor megjelenik a kvíz ablak, ahol egy kvízre választ kell adni.

4, Ha jó választ adott, akkor a blokkoló és a blokkolt góc feloldódik, ha rossz választ adott, akkor újra próbálkozhat.

### 3.5.7 Ajtónyitó gócok



Az ajtónyitó gócok a mellettük lévő ajtókat irányítják. Ha kapnak áramot, akkor kinyitják az ajtót, ellenkező esetben csukva maradnak. Nem minden ajtóhoz tartozik ajtónyitó, szóval nem minden ajtó nyitható.

### 3.5.8 Szint teljesítése

Amikor a felhasználó sikeresen áramot juttat a cél gócba, a játék megáll, és megjelennek a pályán elért statisztikái. Ilyenkor látható a választott nehézség, a használt lépések száma, és a szint megoldására használt idő. Ezek az adatok a ranglistába is bekerülnek. Ha a teljesített szint a felhasználó által elérhető legmagasabb szint, akkor feloldódik a következő szint. Ezután vissza lehet lépni a főmenübe.

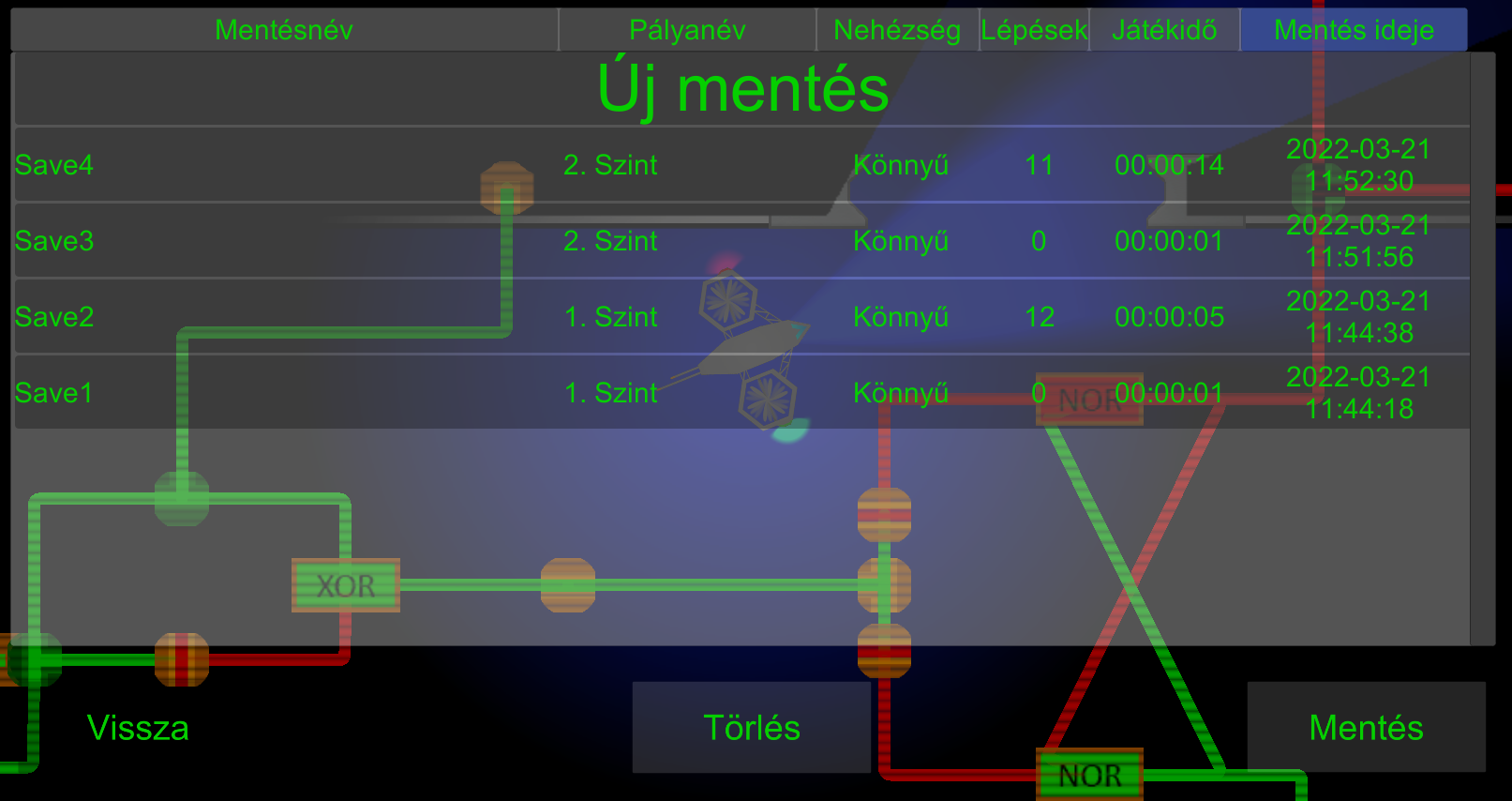
### 3.5.9 Szünet menü



A felhasználó a játékot az Esc gomb megnyomásával bármikor megállíthatja. Ilyenkor megjelenik a szünet menü, és megáll a stopper is. Ebből a menüből visszaléphetünk a főmenübe, vagy akár ki is léphetünk a játékból. Ezek mellett lehetőségünk van mentésre és betöltésre is.

## 3.6 Egyéb funkciók

### 3.6.1 Mentés

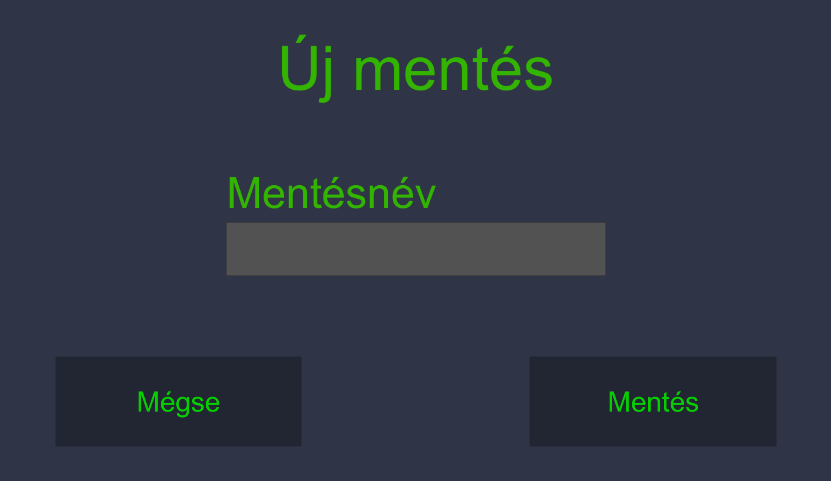
 A mentés ablakot a szünet menüből lehet elérni, itt a felhasználó láthatja a saját mentéseit. Egy mentésre kattintva ki lehet választani azt, és a „Mentés” gombra kattintva felül is lehet írni. Dupla kattintással a mentés azonnal felül írható, a „Mentés” gombra kattintás nélkül.

Mielőtt a mentés felül írásra kerülne, megjelenik egy felugró ablak, ami megkérdezi, hogy biztosak vagyunk-e a döntésünkben.



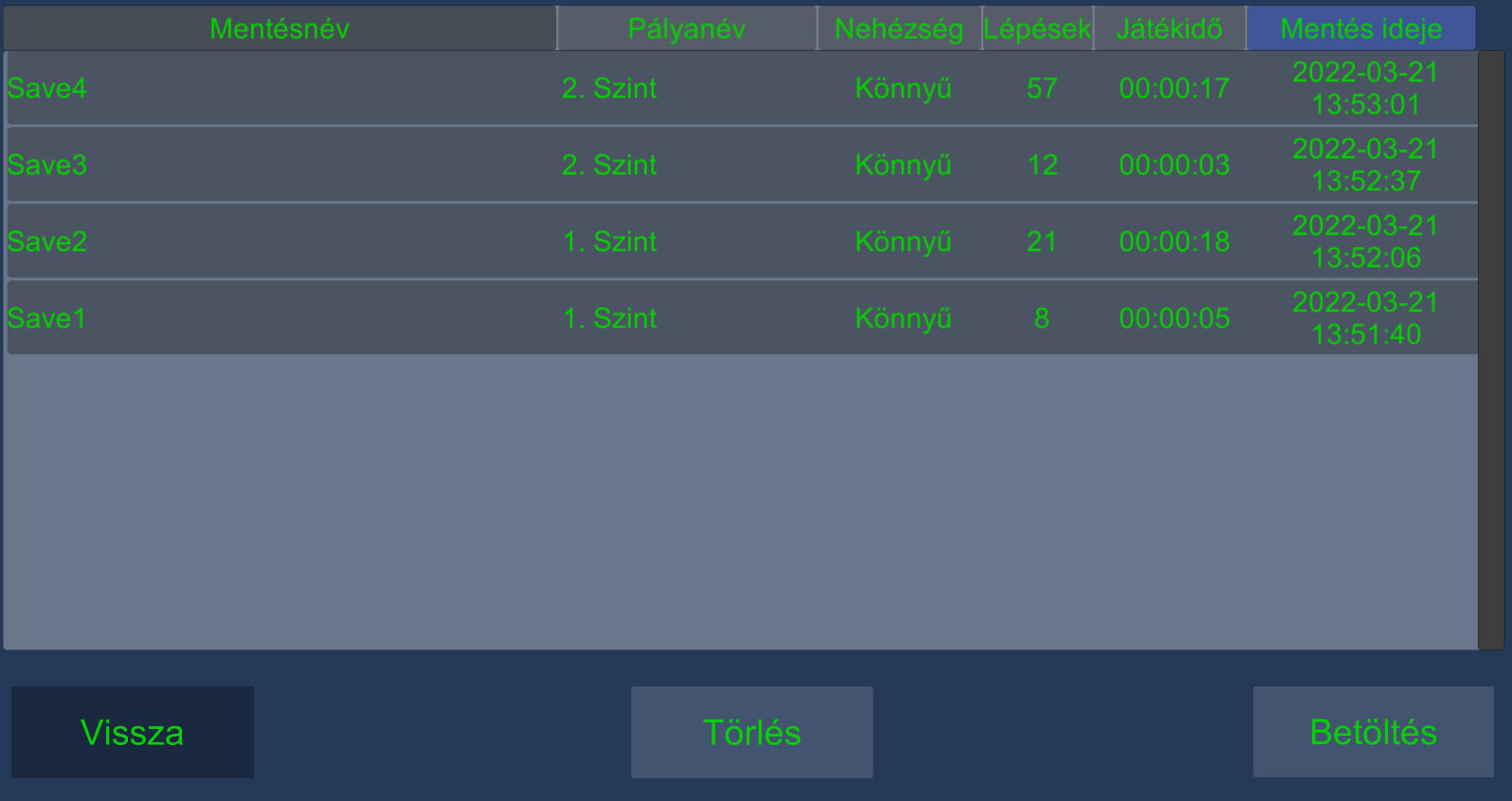
Ha a „Felülírás” gombra kattintunk, akkor felülíródik a mentésfájl, és a mentés adatai az adatbázisban is frissülnek.

Az „Új mentés” gombra kattintáskor megjelenik egy felugró ablak, ahol megadhatjuk az új mentésünk nevét. A „Mentés” gombra kattintáskor a szint pillanatnyi állapota új mentésfájlba íródik, és az adatbázisba új rekord kerül.

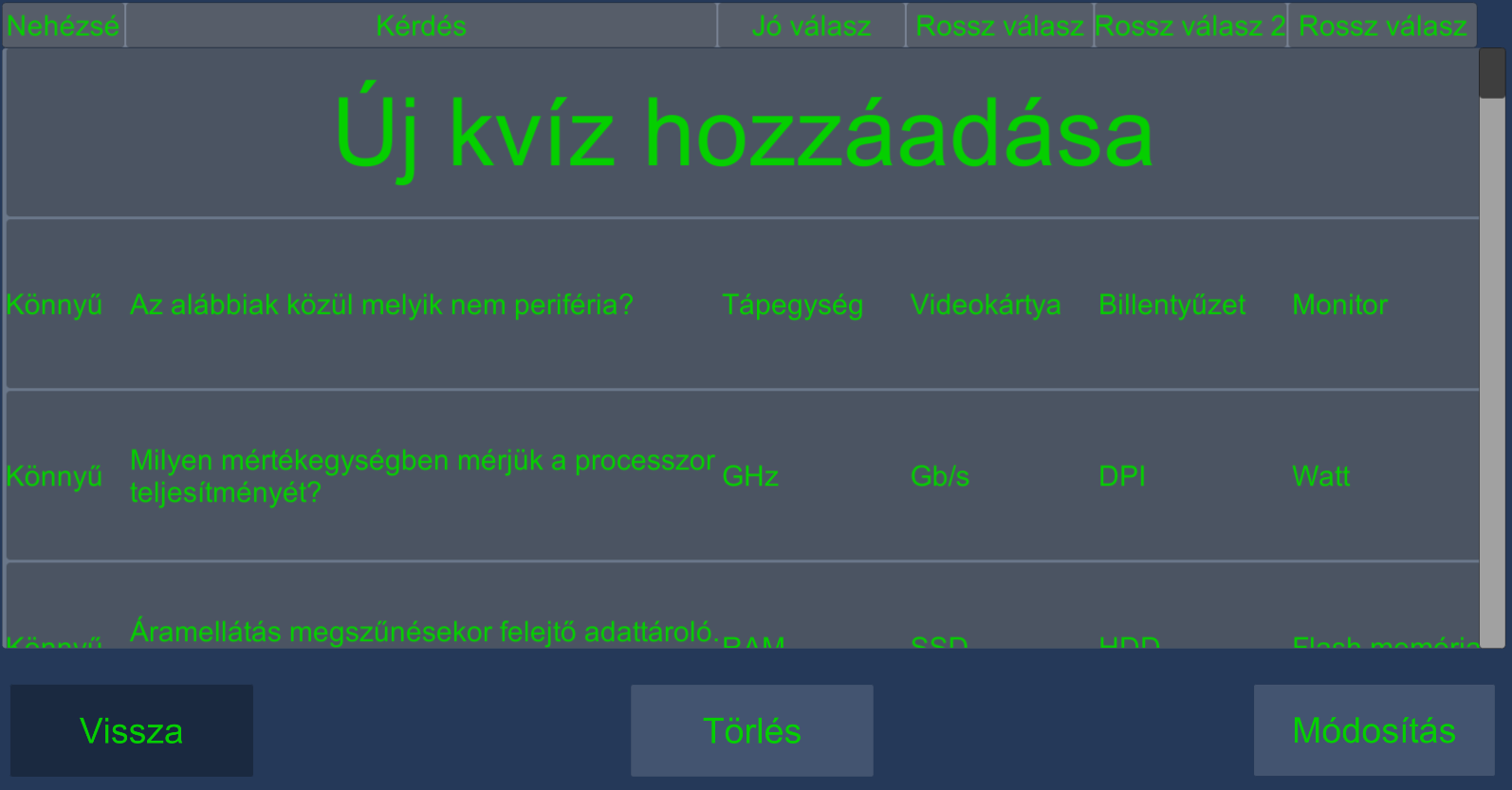


### 3.6.2 Betöltés

A betöltés ablakot a szünet menü „Betöltés”, vagy a főmenü „Játék folytatása” gombjára kattintva lehet elérni. Itt egy mentésre dupla kattintással, vagy egy mentés kiválasztása után a „Betöltés” gombra kattintva tölthetünk be egy mentett játékállást.



### 3.6.3 Kérdések módosítása és hozzáadása

A főmenüben a „Kérdések” gombra kattintva nyithatjuk meg a kvíz ablakot. Itt lehetőségünk van kvízeket hozzáadni és módosítani. 

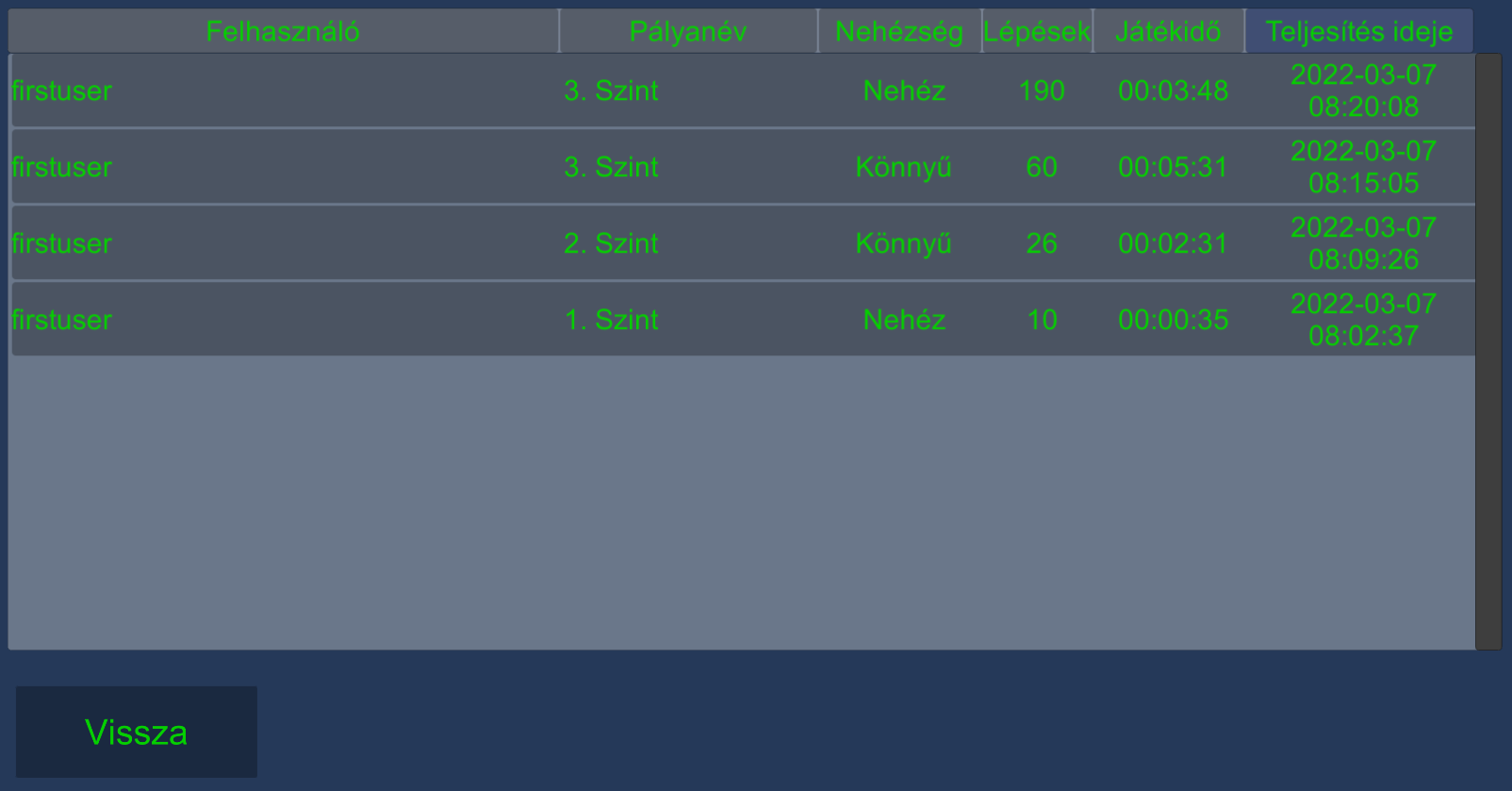
Az „Új kvíz hozzáadása” gomb megnyomásakor megjelenik a kvíz hozzáadó ablak. A szükséges mezők helyes kitöltése után a „Hozzáad” gombra kattintva az új kvíz bekerül az adatbázisba, és az elkövetkező játékmenetekben sorsolható lesz.



Kvíz módosításához először ki kell választanunk egy kvízt, majd a „Módosítás” gombra kattintva megjelenik az előbb említett ablak, és a választott kvíz adatai bekerülnek a beviteli mezőkbe. A „Mentés” gombra kattintáskor elmentődnek a változtatások.

### 3.6.4 Ranglista

A ranglista a főmenüből érhető el, segítségével a felhasználók megtekinthetik egymás eredményeit.



# 4. Összefoglalás

## 4.1 Szakdolgozat célja

A szakdolgozatom célja egy interaktív, több nehézségi szinttel és pályával rendelkező logikai fejtörő és kvízjáték volt. A játék pillanatnyi állapotának mentését, majd annak későbbi betöltését is meg szerettem volna oldani.

## 4.2 Megvalósítás

### 4.2.1 Puzzle rendszer

Legelőször is meg kellett terveznem, hogy egyáltalán hogyan fog működni a logikai puzzle rendszer, és a kvízeket hogyan fogom értelmesen belefűzni ezekbe. A puzzle rendszerhez már voltak ötleteim, a Watch Dogs nevű játék hack puzzle rendszeréből sok inspirációt merítettem, a kvízeket pedig blokkolók beiktatásával integráltam. A gócok, blokkolók és csatlakozók működésének megtervezése után le is kódoltam azokat, és ezek a játékelemek ezután nem is változtak sokat.

### 4.2.2 Karakter kontroller alapjai

A következő dolog a játékos kontroller volt. Egyszerű WASD mozgást implementáltam, és a játékos karaktert a mozgás vektorjának irányába fordítottam. Ez a class a fejlesztés során még sokszor változtatásra került.

### 4.2.3 Kvíz rendszer

Ezután megcsináltam a kvíz rendszert. Ehhez adatbáziskezelő és kvízkezelő osztályokat kellet írnom, és az adatbázis is létre kellet hoznom, hogy legyen miből kérdéseket olvasni.

### 4.2.4 Mentés és betöltés alapjai

A kvíz rendszer elkészítése után megcsináltam még a mentés és betöltés rendszer alapját. Csak egy mentésfájl volt, és a megfelelő gombok megnyomásával lehetett menteni és betölteni.

### 4.2.5 Felhasználókezelés

A regisztrációs és belépés rendszer működésének kezelésére írtam egy főmenü menedzsert, az adatbázis és az adatbáziskezelő funkcionalitását kiterjesztettem felhasználók mentésére és kezelésére is. Ennél a lépésnél a legtöbb munkát maga a felhasználói felület elkészítése és annak elemeinek kezelésének logikájának megvalósítása jelentette.

### 4.2.6 Dinamikus lista menü

Szükségem volt egy dinamikus lista felületre, ami több listafajta megjelenítésére is képes, és az adatokat az adatbázisból veszi. E mellet célom colt az, hogy ehhez a listához adatokat lehessen hozzáadni, valamint azokat módosítani és törölni is lehessen, ahol ezekre a funkciókra is szükség van. Az ezen elvárások szerint elkészített felület és annak kezelő osztálya felelős a mentés, betöltés, ranglista és kvíz böngésző menük funkcionalitásáért. Az összes megemlített menü fajta ugyan azt a felületet használja. Amikor váltunk közöttük, akkor valójában ugyan az az ablak jelenik meg mindegyiknél, csak a kontroller a lista típus szerint jeleníti meg a fejlécet és adatokat. A mentések készítését és betöltését is erre a felületre mozgattam, de a folyamat elvégzéséért nem ez felel.

### 4.2.7 Mentés és betöltés a dinamikus lista menüből

A mentés menü első eleme átlagos lista elem helyett egy „Új mentés” gomb. Megcsináltam, hogy erre a gombra kattintáskor feljöjjön egy mentés ablak, ahonnan kezdeményezhetünk egy új mentést. Ugyanekkor implementáltam a mentések felülírását és törlését is. Ezután megoldottam, hogy ezeket a mentéseket betöltés céljából is ki lehessen listázni.

### 4.2.8 Megvilágítás

Az alap 2D megvilágítást leváltottam a Unity URP-re, elsötétítettem a pályákat, majd fényeket tettem a drónra. Írtam még egy class-t, amely a falak SpriteShape-jéhez idomítja azok ShadowCaster2D-jeit.

### 4.2.9 Hologram shader

A pályák elsötétítése azt is jelentette, hogy a gócok és azok csatlakozói sem láthatók a sötétben. A puzzle-ok átláthatósága pedig eléggé fontos dolog, így úgy döntöttem, hogy erre csinálok egy megvilágítatlan shadert. Ezzel elértem, hogy a fontos játékelemek mindig láthatók legyenek, és egy folyamatosan gördülő maszk miatt úgy tűnik, mintha hologramokkal lennének láthatóvá téve.

### 4.2.10 Dinamikus lista rendezés

Megcsináltam, hogy a lista egy fejléc elemére kattintáskor a lista elemeit a kiválasztott oszlop adatai szerint rendezi.

### 4.2.11 Kód dokumentáció

A játék fejlesztése eljutott egy olyan stádiumba, ahol a kód már java részt készen volt, nagy modifikációk nem voltak várhatóak. A létrehozott osztályokat és azok metódusait xml címkék segítségével dokumentáltam, és a kódba kommenteket is írtam. Az imént említett xml címkékből a doxygen nevű programmal weboldalas alapú dokumentációt tudok generáltatni.

### 4.2.12 Adatbázis feltöltése kérdésekkel

A kvízek kitalálása meglepően időigényes volt. Mindhárom nehézséghez 20-20 kérdést tettem az adatbázisba.

### 4.2.13 Pályák elkészítése

A játékba végül három szint került be, és azoknak a gyártása is több időbe telt, mint amire számítottam. Az első szint elkészítése aránylag könnyű volt, mivel azt csak egy egyszerű bemutatónak szántam, és nem egy komplexebb kihívásnak.

A második szint egy próba pályából lett átalakítva, ahhoz csak hozzá kellett adnom valamennyit, az alap ötlet már megvolt.

A harmadik szint már más téma volt. Egy nagyobb pályát szerettem volna csinálni, és ehhez se egy félkész pályám, se alap ötletem nem volt. Elkezdtem pályaelrendezéseket megrajzolni, és azt megtervezni, hogy azon hogyan kéne végig menni. Néhány iteráció után létrehoztam azt az elrendezést, ami végül a harmadik szint elrendezése lett. A puzzle is megtervezhető lett volna hasonló módon, de az kevésbé lett volna előnyös. Könnyebbnek találtam azt, hogy egyszerre csak egy kis részt terveztem és csináltam meg, és azon addig változtattam, iteráltam, amíg jónak nem tűnt.

## 4.3 Fejlesztési lehetőségek

### 4.3.1 Kinézet fejlesztése

A játék készítésekor a kinézetre szinte semmi figyelmet nem fordítottam. A vizuális élmény fejlesztése megoldható lenne textúrák használatával és dekoratív elemek elhelyezésével a pályákon. Emellett a felhasználói felület is új kinézetet kaphatna.

### 4.3.2 Hangok hozzáadása

A játékban jelenleg egy hang sincsen. Amikor a játékos például elforgat egy gócot, vagy rákattint egy gombra, akkor a játék visszajelzésként lejátszhatna hangokat. Egy kis háttérzene sem ártana.

### 4.3.3 Több pálya és kérdés hozzáadása

Jelenleg elég kevés kérdés és pálya van a játékban, ezért hamar kijátszható, megunható. Több tartalom hozzáadásával ez orvosolható lenne.

### 4.3.4 Adaptálás mobil eszközökre

A játékot egy joystick és egy menü gomb hozzáadásával könnyen érintőképernyőbaráttá lehetne tenni. Jelenleg, ha a játékos egy kvízre rossz választ ad, akkor a lépésszámláló megnövekszik eggyel. Ehelyett be lehetne vezetni egy élet rendszert. A játékos kezd x számú élettel, és ha egy kvízt rosszul válaszol meg, akkor levonunk egyet. Ha elfogyott az összes élete, akkor nem oldhat fel blokkolókat. Az életek egyesével y percenként regenerálódhatnának. Ezzel a rendszerrel be lehetne vezetni azt, hogy a játékos egy hirdetés megtekintésével életeket kaphasson vissza, és folytathassa a játékot. A játékot ezután fel lehetne tölteni a Play Áruházba, hogy bevételt generáljon.

### 4.3.5 Online belépés

Több értelme lenne, hogyha a lokális fiókok helyett online történne a bejelentkezés. Ez arra is adna lehetőséget, hogy egy szerver kövesse azt, hogy a felhasználók hányadik szinten tartanak.

### 4.3.6 Online ranglista

Egy online szerver segítségével megoldható lenne egy globális ranglista létrehozása.

### 4.3.7 Kvízek online megosztása

Megoldható lenne, hogy a felhasználók saját kvízgyűjteményeket csinálhassanak, és ezeket másokkal megosszák. Lenne egy kérdéscsomag böngésző, ahonnan kérdéscsomagokat tölthetnének le a felhasználók. Ehhez egy értékelés rendszer is beiktatható. A fűmenüben ezután lenne még egy menüpont, ahol ki lehetne választani, hogy a letöltött csomagok közül melyikek legyenek aktívak.

### 4.3.8 Pályaszerkesztő hozzáadása

Egy pályaszerkesztő hozzáadásával lehetővé tehetném, hogy a játékosok saját pályákat készítsenek. Ezeket a pályákat ezután a kvízek megosztásához hasonló rendszerrel megoszthatnák egymással.

### 4.3.9 Nyelvi lokalizáció

A játék jobb elérhetőségéért szükséges egy angol fordítás hozzáadása. A lokalizációt ezután ki lehetne terjeszteni több nyelvre is. A nyelvek között egy nyelvválasztó menüvel lehetne majd váltani.

# 5. Irodalomjegyzék

* <https://docs.unity3d.com/>
* <https://docs.microsoft.com/>
* <https://stackoverflow.com/>
* <https://www.w3schools.com/>
* <https://www.youtube.com/channel/UCSaBocKAyT5MIQlz4nn_y0g>
* <https://www.youtube.com/c/Brackeys>
* <https://www.youtube.com/c/CodeMonkeyUnity>