Békéscsabai SZC Nemes Tihamér Technikum és Kollégium

Szakképesítés megnevezése: Szoftverfejlesztő

Azonosító száma: 54 213 05

**Off time**

Konzulens Neve Készítette: Varga András Bendegúz

Gajda Judit 5/13.B

Oktatási azonosító: 72169956207

Békéscsaba, 2021/2022

Tartalom

[1. Bevezetés 3](#_Toc98317992)

[1.1 Köszönetnyilvánítás 3](#_Toc98317993)

[1.2 Indoklás 3](#_Toc98317994)

[2. Fejlesztői dokumentáció 4](#_Toc98317995)

[2.1 Fejlesztői környezet 4](#_Toc98317996)

[2.1.1 Programozási nyelv és környezet 4](#_Toc98317997)

[2.1.2 Játékmotor 4](#_Toc98317998)

[2.2 Adattárolás 5](#_Toc98317999)

[2.2.1 Adatbázisszerkezet 5](#_Toc98318000)

[2.2.2 A users tábla 5](#_Toc98318001)

[2.2.3 A saves tábla 5](#_Toc98318002)

[2.2.4 A wins tábla 5](#_Toc98318003)

[2.2.5 A questions tábla 6](#_Toc98318004)

[2.2.6 Fájlrendszer 6](#_Toc98318005)

[2.2.7 Mentés menete 6](#_Toc98318006)

[2.2.8 Betöltés menete 7](#_Toc98318007)

[2.3 Adatbázis kezelés 8](#_Toc98318008)

[2.3.1 Regisztráció és bejelentkezés 8](#_Toc98318009)

[2.3.2 Mentés 8](#_Toc98318010)

[2.3.3 Betöltés 9](#_Toc98318011)

[2.3.4 Szintteljesítés 9](#_Toc98318012)

[2.3.5 Ranglista 9](#_Toc98318013)

[2.3.6 Kvíz kérése 10](#_Toc98318014)

[2.3.7 Kvízek megtekintése, módosítása 10](#_Toc98318015)

[2.4 Játékelemek 10](#_Toc98318016)

[2.4.1 Játékos drón 10](#_Toc98318017)

[2.4.2 Energiagócok 11](#_Toc98318018)

[2.4.2.1 Forgatható gócok 11](#_Toc98318019)

[2.4.2.2 Logikai gócok 11](#_Toc98318020)

[2.4.2.3 Blokkoló gócok 11](#_Toc98318021)

[2.4.2.4 Cél góc 11](#_Toc98318022)

[2.4.2.5 Ajtónyitó 12](#_Toc98318023)

[2.4.2.6 Ajtó 12](#_Toc98318024)

[2.4.2.7 Csatlakozó 12](#_Toc98318025)

[2.4.3 Kvíz ablak 12](#_Toc98318026)

[2.4.4 Szünet ablak 13](#_Toc98318027)

[2.5 Tesztelés 13](#_Toc98318028)

[2.5.1 Tesztelés módja 13](#_Toc98318029)

[2.5.2 Talált hibák 13](#_Toc98318030)

[2.5.3 Egyéb megjegyzések, visszajelzések 14](#_Toc98318031)

# 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 program 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 az összes többi 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 nevén és hashelt jelszaván 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 mentésmappát a felhasználó AppData/LocalLow mappán belüli saját magának létrahozott mappájában tárolja. Az összes felhasználó mentéseit ugyanebben a mentés mappában tárolja, és ezeket az adatbázis saves táblájának segítségével kezeli.

### 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, 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 visszatöltetjuk 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 objektumban.

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 objektumba, 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 Fefresh 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 drón

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.

### 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.

### 2.5.3 Egyéb megjegyzések, visszajelzések

Többen is megjegyezték, hogy a mozgás reszponzív volt, és a drón forgása gyors volt, de egyenletes. Többen is 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.