4. Adatbázis terv

Dokumentum verzió: 1.0

Kiadás dátuma: 2025.11.16

Készítette: Infinite Loopers fejlesztőcsapat

1. Bevezetés

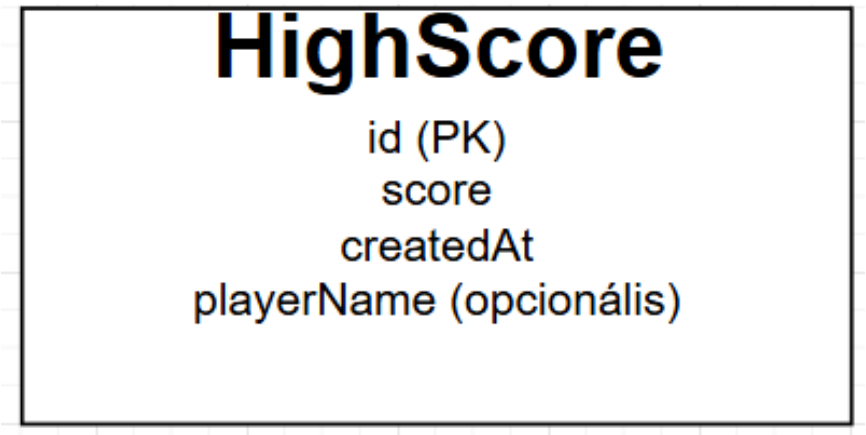
Az Infinite Loopers játék nem használ hagyományos relációs adatbázist (pl. MySQL, PostgreSQL), mivel a rendszer célja egy egyszerű, offline működő, egyjátékos endless runner élmény biztosítása. A játék mindössze néhány olyan adatot kezel, amelyek tartós tárolása szükséges: elsősorban a high score értéket, illetve opcionálisan bizonyos beállításokat (hangerő, felbontás). Ezek mennyisége és struktúrája nem indokolja egy teljes adatbázis-rendszer bevezetését.

A rendszer ezért fájlalapú adatkezelést alkalmaz: a high score értéke egy egyszerű szöveges fájlban kerül eltárolásra (pl. highscore.txt), amelyet a ScoreSystem modul olvas és ír. Az adatbázis-terv ebben a dokumentumban ennek a minimális adattárolási megoldásnak a logikai modelljét, relációs megfeleltetését és az adatkezelési folyamatokat mutatja be, hogy a későbbi esetleges adatbázisra váltás is könnyen végrehajtható legyen.

2. ER-diagram

A logikai adatmodell központi entitása a Highscore, amely a játékos által elért legmagasabb pontszám adatainak tárolására szolgál. Bár a jelenlegi implementáció szöveges fájlt használ, a későbbi bővíthetőség érdekében az entitást úgy definiáljuk, mintha adatbázisban kerülne tárolásra. Így a high score rekord kiegészíthető további metaadatokkal, például mentési dátummal vagy játékosnévvel.

Az alábbi ER-diagram bemutatja a Highscore entitást és annak fő attribútumait:



A Highscore entitás az alábbi attribútumokat tartalmazza:
- id: egyedi azonosító (elsődleges kulcs), amely egy high score rekordot azonosít;
- score: az elért pontszám egész számként;
- createdAt: a rekord mentésének dátuma és időpontja;
- playerName: opcionális mező, amennyiben a későbbiekben játékosnév rögzítése is szükségessé válik.

3. Relációs modell

A relációs modell a logikai ER-diagram táblaszintű leképezése. Bár jelenleg tényleges adatbázis nem kerül használatra, a Highscore entitás relációként az alábbi módon értelmezhető:

Highscore(id, score, createdAt, playerName)

- id: INTEGER, PRIMARY KEY – egyedi azonosító minden rekordhoz;
- score: INTEGER, NOT NULL – a játékos által elért pontszám;
- createdAt: DATETIME, NOT NULL – a mentés időbélyege;
- playerName: VARCHAR, NULL – opcionális játékosnév.

Amennyiben a rendszer a jövőben további adatokat tárolna (például több high score rekordot, vagy külön beállításokat), bevezethető egy Settings tábla is, amely kulcs–érték párokat tartalmazna:
Settings(key, value)
ahol a key például 'musicVolume', 'sfxVolume' vagy 'resolution' lehetne. Jelen implementációban ezek az adatok szintén egyszerű konfigurációs fájlban tárolhatók.

4. Adatkezelés folyamata

A játék indulásakor a ScoreSystem megpróbálja beolvasni a highscore.txt fájlt. Ha a fájl létezik és érvényes adatot tartalmaz, akkor a benne tárolt pontszámot tekinti aktuális high score-nak; ellenkező esetben a high score értéket 0-ra inicializálja. A játék futása közben a ScoreSystem folyamatosan nyilvántartja az aktuális pontszámot, és összeveti az elért eredményt a high score-ral.

A játék végén, amikor a Player egy akadállyal ütközik és a GameManager Game Over állapotba lép, a ScoreSystem ellenőrzi, hogy az aktuális pontszám nagyobb-e, mint a korábban eltárolt high score. Amennyiben igen, frissíti a highscore.txt fájl tartalmát, ezzel felülírva a korábbi rekordot. A mentés tipikusan egyszerű szöveg formájában történik, egyetlen sorban tárolva a pontszámot, esetleg további mezőkkel kiegészítve (dátum, játékosnév).

A lekérdezés, mentés és módosítás így mind fájlműveleteken keresztül valósul meg, amelyek adott esetben könnyen lecserélhetők adatbázis-hozzáférésre (például SQL-lekérdezésekre), anélkül, hogy a játék üzleti logikáját jelentősen módosítani kellene. A ScoreSystem homogén felületet biztosít az adatkezeléshez a többi modul (pl. GameManager, UI) számára.

5. Adatkezeléssel kapcsolatban álló osztályok

ScoreSystem: a legfontosabb adatkezelő osztály, amely a high score érték beolvasását, tárolását és frissítését végzi. Olyan metódusokat tartalmaz, mint a loadHighscore(), saveHighscore(score) és updateScore(currentScore), amelyek elrejtik a fájlkezelés részleteit a többi modul elől.

GameManager: bár közvetlenül nem végez fájlműveleteket, szorosan együttműködik a ScoreSystem osztállyal. A játékmenet során ő dönt arról, mikor kell a pontszámot frissíteni, illetve mikor kell a játék végén a high score mentését kezdeményezni. Így a GameManager magasabb szintű üzleti logikát valósít meg az adatkezelés felett.

UI/HUD komponensek: a felhasználói felület elemei (pl. pontszám kijelzés) a ScoreSystem által szolgáltatott adatokat jelenítik meg, de nem módosítják közvetlenül az adattárolást. Ez a szerepkör-szétválasztás biztosítja, hogy az adatkezelésért kizárólag az arra tervezett modulok feleljenek.

Amennyiben a jövőben adatbázis bevezetése válik szükségessé, könnyen létrehozható egy külön HighscoreRepository vagy DataAccess osztály, amely a ScoreSystem mögött elrejti az adatbázis-specifikus műveleteket (SQL lekérdezések, kapcsolatkezelés). Így a jelenlegi fájlalapú megoldás fokozatosan, visszafelé kompatibilis módon váltható át adatbázis-kezelésre.

6. Összefoglalás

Bár az Infinite Loopers jelenlegi verziója nem használ hagyományos relációs adatbázist, az adatbázis-terv elkészítése segít abban, hogy a high score kezelésére vonatkozó megoldás átlátható és bővíthető legyen. A Highscore entitás logikai és relációs modellje, az adatkezelési folyamatok és az érintett osztályok részletes bemutatása biztosítja, hogy a későbbi fejlesztések során akár egy teljes adatbázis-rendszer is könnyen integrálható legyen a játékba.