Beszámoló szakmai gyakorlatról

Miskolci Egyetem

Készítette:

Sebe Zsolt

Programtervező informatikus

Neptun kód: ACC02G

e-mail cím: sebezsolt2@gmail.com

Üzemi instruktor:

Piller Imre

tanársegéd

Alkalmazott Matematikai Intézeti Tanszék, Miskolci Egyetem

telefonszám: +3646/565-111/14-50

Miskolc, 2025

Tartalomjegyzék

1 A feladat bemutatása

A “Túlélés a Szocializmusban” című játék egy túlélés-orientált szimuláció, amely a szocialista korszak mindennapjainak nehézségeit és sajátos társadalmi dinamikáját dolgozza fel. A játék célja, hogy a játékos egy államilag szabályozott, korlátozott lehetőségeket kínáló rendszerben találja meg a túlélés útját – akár törvényes, akár féllegális eszközökkel. A projekt célközönsége a komolyabb hangvételű szimulációs élményeket kereső játékosok, akik érdeklődnek a történelmi ihletésű, morális döntéseket is tartalmazó játékmenetek iránt.

A játék cselekménye egy fiktív, ám a valódi szocialista rendszerek jegyeit idéző világban játszódik. A játékos feladata, hogy saját és családja mindennapi megélhetését biztosítsa egy gazdaságilag nyomott, bizonytalanságokkal teli korszakban, ahol a legkisebb döntés is súlyos következményekkel járhat.

A játékmenet középpontjában a túlélés áll: a játékos folyamatosan mérlegeli, hogyan ossza be idejét és szűkös erőforrásait a munka, a család, valamint a saját testi-lelki állapotának megőrzése között.

2 Technológia kiválasztása

A projekt fejlesztéséhez csapatunk a Godot Game Engine-t választotta. A Godot egy nyílt forráskódú, platformfüggetlen játékmotor, amely támogatja a 2D és 3D játékfejlesztést egyaránt. [1], [2] A motor fő előnyei közé tartozik a könnyű kezelhetőség, a gyors prototípus-készítés lehetősége, valamint a beépített vizuális szerkesztő, amely megkönnyíti a jelenetek és erőforrások kezelését.

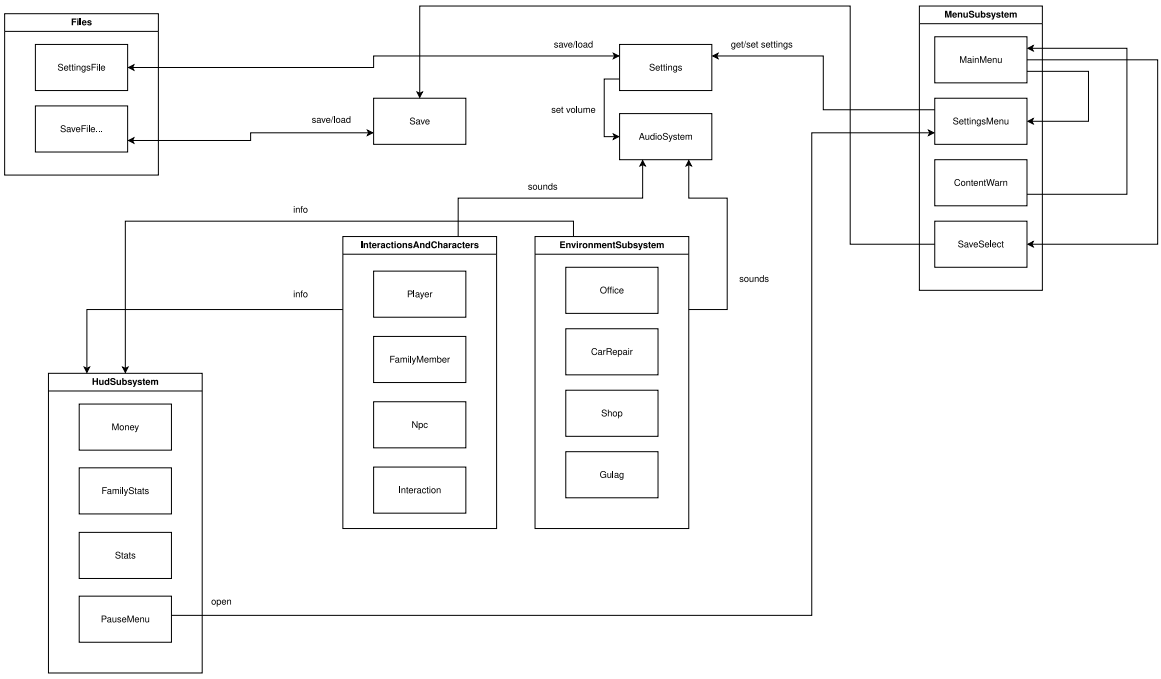
3 A tervezés folyamata

A projekt tervezése során a feladatokat igyekeztünk átláthatóan és egyenletesen elosztani egymás között, ugyanakkor mindenkinek lehetősége volt a saját érdeklődési körének és erősségeinek megfelelően hozzájárulni a közös munkához. A folyamat során a rendszeres megbeszélések és konzultációk biztosították, hogy a dokumentáció és a fejlesztés egységes irányt kövessen.

3.1 Dokumentáció

Az első időszakban főként az adminisztratív feladatokra koncentráltunk: közösen megbeszéltük a kezdeti dokumentáció felépítését, valamint felosztottuk a feladatokat. Ezt követően sor került a verziókövetéshez szükséges eszközök (git, GitHub) és a dokumentációs környezet (MkDocs) beállítására, illetve a projekt weboldalának elkészítésére. Ezekhez a munkákhoz személyesen is hozzájárultam, különösen az MkDocs beüzemelésével (`2534f04`).

A tervezési fázis másik fontos része a közös vízió megalkotása és felosztása volt. Ezekhez aktívan részt vettem a megbeszéléseken, ahol a felmerülő ötletek alapján javaslatokat dolgoztunk ki, majd kidolgoztam a rendszer lehetséges megoldásait és a funkcionális, valamint nem funkcionális követelményeket (`6e34a5a`). Emellett közreműködtem az SRS dokumentum “Bevezetés”, “Áttekintés”, és “Használhatóság” részeinek megírásá-



Ábra 1: A játék fő alrendszereinek komponensdiagramja és azok kapcsolatai

ban (`6dee4b4`), az Ábra 1 elkészítésével, valamint a kezdeti formázási és technikai hibák javításában (`b720b39`).

3.2 Telepítés

A Godot telepítését a `dnf install godot` parancs segítségével végeztem el.

3.3 Fejlesztés

A későbbiekben a hangsúly fokozatosan a gyakorlati fejlesztés előkészítésére helyeződött. Részt vettem a Godot mappastruktúrájának megtervezésében és a scenes rendszer átbeszélésében. Különösen hasznosnak bizonyultak ezek a megbeszélések, hiszen segítettek abban, hogy a későbbi fejlesztési munkák egységes alapokra épüljenek.

A menürendszer mellett a HUD felület, a mentéskezelő és a beállítási rendszer (`0225e11`, `739f471`, `13674a2`, `df1a05d`) kialakítását is én végeztem. Ez utóbbi nemcsak a felhasználói beállítások mentését biztosította, hanem a hangkezelő rendszert is: a program a felhasználó által választott hangerőszinteket elmentette, majd automatikusan a megfelelő Godot audióbuszokra állította be, biztosítva a játék testreszabható és konzisztens hangélményét.

A projekt keretében sajnos nem sikerült a játék teljes funkcionalitását megvalósítani, így a fejlesztett modulok jelenleg csak a demó jellegű részeket tartalmazzák. Ennek ellenére a kialakított rendszerek stabil alapot nyújtanak a további bővítéshez és teszteléshez, valamint a fejlesztési folyamat során szerzett tapasztalatok értékesek a jövőbeli munkák szempontjából.

4 Implementáció

4.1 Mentési Rendszer

A játékon belüli mentések kezelésére saját mentési rendszert dolgozotunk ki, amelynek megvalósítását én vállaltam (4a2df1d). A megoldás alapja két függvény:

```gdscript
func save_state(
    ns: StringName,
    state: Dictionary[StringName, Variant]
) -> Error

func load_state(ns: StringName) -> Dictionary
```

Az `ns` paraméter egyfajta névtérként (namespace) működik, így a játék különböző részei elkülönítve tárolhatják az állapotukat. A `load_state()` visszaadja az adott névtérhez korábban elmentett adatokat, vagy üres szótárat, ha még nincs tárolt állapot.

A mentések atomikusan történnek: először ideiglenes fájlba írjuk az adatokat, majd átnevezés után válnak érvényessé. Így biztosítható, hogy vagy a teljes új mentés, vagy a korábbi állapot maradjon meg, elkerülve a fájlok részleges korruptálódását.

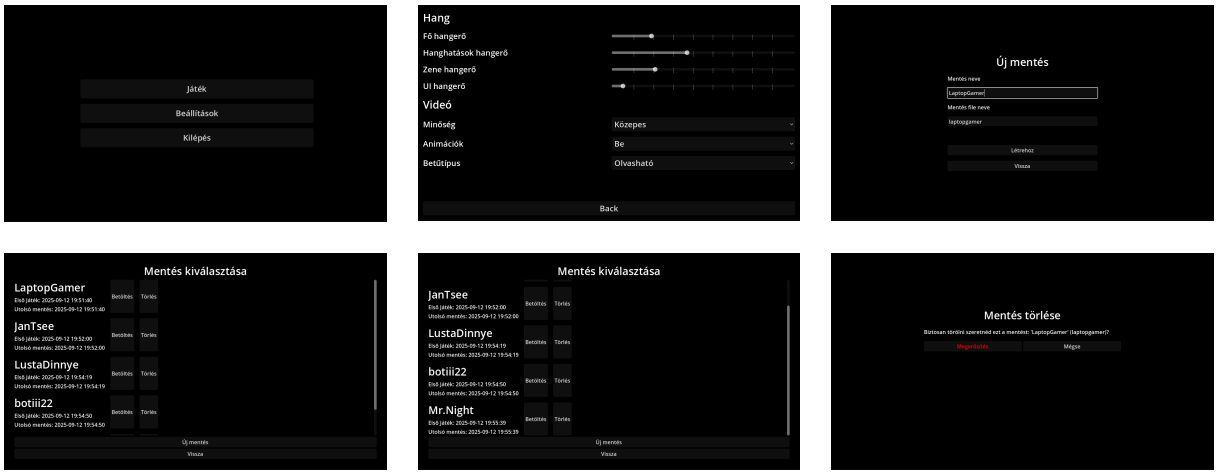
A mentések JSON formátumban kerülnek tárolásra. A legfelső szint egy objektum, amelyben minden kulcs a megfelelő `ns` névtérnek felel meg. A `__meta` egy speciális kulcs, amely a mentéshez tartozó metaadatokat (rögzítés ideje, utolsó módosítás időpontja, felhasználónév stb.) tartalmazza.

```json
{
  "__meta": {
    "ctime": "2025-09-12 19:51:40",
    "mtime": "2025-09-12 19:56:09",
    "name": "LaptopGamer"
  },
  "shop": {
    "correct_answers": 0
  }
}
```

4.2 Menürendszer

A projekt menürendszerét teljes egészében én valósítottam meg a Godot-ban. A rendszer a játék különböző felületeit kezeli, beleértve a főmenüt, a beállításokat, a mentések kezelését és a figyelmeztető üzeneteket.

A struktúra a következőképpen épült fel:



Ábra 2: A játék menürendszerének fő nézetei, beleértve a főmenüt, a mentéskezelőt és a beállítási lehetőségeket

Ábra 3: A játék futás közben megjelenő HUD felülete Piotr (játékos) és Mihalina (családtag) karakterrel

```
/scenes/menu/
├── hud_entry_family.{gd,tscn}
├── hud_entry_player.{gd,tscn}
├── hud.{gd,tscn}
├── main_menu.{gd,tscn}
├── new_save_menu.{gd,tscn}
├── save_delete_confirm.{gd,tscn}
├── save_entry.{gd,tscn}
├── save_select_menu.{gd,tscn}
├── settings_menu.{gd,tscn}
└── warn_menu.{gd,tscn}
```

A menük logikáját GDScript-ben implementáltam, a különböző jelenetek érintkezését és a felhasználói interakciók kezelését biztosítva. A menürendszer tartalmazza a mentések kiválasztását és létrehozását, a beállítások módosítását, valamint a különböző HUD elemeket, melyek dinamikusan frissülnek a játék állapota szerint. A menürendszer vizuális felépítését és egyes nézeteit az Ábra 2 és Ábra 3 szemlélteti.

4.2.1 Menürendszer részletesen

Az egyes elemek a következő feladatokat látják el:

- `hud_entry_family.{gd,tscn}`: Egy családhoz tartozó HUD elemek megjelenítése, az aktuális tag állapotának vizualizálása.
- `hud_entry_player.{gd,tscn}`: A játékos karakterének állapotát mutatja a HUD-on, beleértve életpontokat és egyéb státuszokat.
- `hud.{gd,tscn}`: A teljes HUD menedzsmentjét végzi, összehangolja az összes HUD-elem frissítését.
- `main_menu.{gd,tscn}`: A főmenüt valósítja meg, innen lehet új játékot indítani, betöltést kezdeményezni, vagy a beállításokat elérni.
- `save_delete_confirm.{gd,tscn}`: Mentés törlésének megerősítő párbeszédpanelje, biztonsági ellenőrzéssel.
- `save_entry.{gd,tscn}`: Egyetlen mentés bejegyzésének reprezentációja a mentésválasztó menüben.
- `save_select_menu.{gd,tscn}`: A mentések listázását és kiválasztását biztosító felület.
- `settings_menu.{gd,tscn}`: A játék beállításait kezelő felület, amely a hang-, grafikai és egyéb konfigurációk módosítását teszi lehetővé. A menü a `GameSettings` globális változó segítségével kezeli a beállításokat, és biztosítja azok betöltését és mentését. A hangbeállítások (master, SFX, zene, UI) a megfelelő audióbuszokra kerülnek alkalmazásra, így a változtatások azonnal érvényesülnek. A grafikai opciók között a videó minősége, animációk engedélyezése és betűtípus választás szerepel. A “Back” gomb a módosítások mentése után visszatér a szülő menübe.
- `new_save_menu.{gd,tscn}`: Az új mentés létrehozására szolgáló felület. A játékos itt megadhatja a mentés nevét és fájlazonosítóját, amelyeket a rendszer automatikusan megtisztít (szóközök, kis-/nagybetűk kezelése) és ellenőriz egy reguláris kifejezés alapján. A menü hibakezelést is biztosít: üres név vagy fájlnév esetén, érvénytelen fájlnév használatakor, illetve már létező mentés esetén a felület hibaüzeneteket jelenít meg. Sikeres létrehozáskor a `SaveManager` új mentést hoz létre, és a szülő menü értesítést kap a lista frissítéséről.

4.3 Beállításkezelő Rendszer

A `GameSettings` modul felelős a játék globális konfigurációs paramétereinek kezeléséért, ideértve a hang-, grafikai- és nyelvi beállításokat. A modul biztosítja, hogy a felhasználó által végrehajtott változtatások a menün keresztül vagy manuálisan a `settings.ini` fájl szerkesztésével is érvényesüljenek.

A beállítások mentése atomikusan történik, pont úgy mint a mentéseknél.

A modul felel a hangbeállítások alkalmazásáért is: a felhasználó által megadott hangerőszinteket automatikusan a megfelelő Godot audióbuszokra állítja.

A `GameSettings` a Táblázat 1 által felsorolt beállításokat kezeli.

A `content-warn-ack` beállítás különleges jelentőséggel bír: a játék indításához a tartalomfigyelmeztetés elfogadása kötelező. Ha a felhasználó még nem fogadta el, a játék

Táblázat 1: A játék `settings.ini` fájljában tárolt beállítási lehetőségei

Név	Típus	Leírás	Alapértelmezett
`[volume]`		Hangerő szabályok	
`master`	`int [0, 100]`	Master hangerő	`40`
`sfx`	`int [0, 100]`	Effektusok hangereje	`100`
`music`	`int [0, 100]`	Háttérzene hangereje	`100`
`ui`	`int [0, 100]`	UI hangereje	`100`
`[video]`		Grafikai beállítások	
`quality`	`low, medium, high`	Grafika minőség	`medium`
`animations`	`bool`	UI animációk engedélyezése	`true`
`font`	`pixel, readable`	Betűtípus	`pixel`
`[main]`		Egyéb beállítások	
`lang`	`string`	Játék nyelve	`ask`
`content-warn-ack`	`bool`	Tartalom figyelmeztetés elfogadva	`false`

indításakor a figyelmeztetés megjelenik, és elfogadása után többé nem kerül bemutatásra.

5 Tesztelés

A projekt során a csapat kiemelt figyelmet fordított a tesztelésre, hogy a fejlesztett rendszerek megbízhatóan működjenek, és a hibák korán felderítésre kerüljenek. A tesztek több szinten valósultak meg, beleértve a funkcionális ellenőrzéseket, az integrációs teszteket és az állapotmentési rendszer ellenőrzését.

5.1 Állapotmentési rendszert ellenőrző tesztek

Az általam készített tesztek a `SaveManager` működését vizsgálták (a75a868). A tesztek célja az volt, hogy ellenőrizzük a mentések létrehozását, betöltését, az adatok helyes tárolását és a fájlok törlését. Ezek a tesztek biztosították, hogy a mentések atomikusan történjenek, és a betöltött adatok megegyezzenek a mentéskor tárolt értékekkel. A tesztek a projekt forrásában a `tests/test_save_manager.gd` fájlban találhatók.

A tesztek főbb ellenőrzéseit a Táblázat 2 tartalmazza.

5.2 Egyéb tesztek

Az általam készített `SaveManager` teszteken kívül a csapat többi tagja is írt automatizált teszteket, amelyek a játék különböző moduljainak működését ellenőrzik. Ezek főbb jellemzőit a Táblázat 3 foglalja össze.

5.2.1 Atomikus mentés ellenőrzése (manuális teszt)

Ez a teszt biztosítja, hogy a `SaveManager` által írt fájlok atomikusan frissülnek: az adatokat először egy ideiglenes fájlba írja, majd átnevezéssel cseréli le a végleges fájlt.

Táblázat 2: A SaveManager főbb funkcióit ellenőrző automatizált tesztek

test_save_create	Ellenőrzi, hogy a SaveManager képes-e új mentésfájlokat létrehozni, és hogy a fájlok valóban megjelennek a mentési könyvtárban.
test_save_meta_exists	Ellenőrzi, hogy a mentésekhez tartozó metaadatok, például a mentés neve és fájlazonosítója, helyesen kerülnek-e nyilvántartásra, és lekérhetők-e a SaveManager listázó függvényével.
test_save_load_file	Teszteli, hogy egy meglévő mentésfájl sikeresen betölthető, és a SaveManager a megfelelő státuszkóddal tér vissza.
test_store_and_load_data	Ellenőrzi, hogy a különböző névterek (namespace) használatával tárolt adatok pontosan visszaállíthatók legyenek, beleértve a számokat és listákat is.
test_save_erase	Ellenőrzi, hogy a mentések fájljai valóban törlődnek, és a SaveManager nem tartja nyilván a törölt mentéseket.

Táblázat 3: Összefoglaló a játék moduljait ellenőrző egyéb automatizált tesztekről

test_family.gd	Ellenőrzi a Family objektum létrehozását, a get_description() metódus működését, valamint a getter és setter függvények helyes működését a családtag adatok kezeléséhez.
test_player.gd	Ellenőrzi a Player objektum létrehozását, a get_description() metódust, valamint a játékos attribútumok getter és setter függvényeinek helyességét, ideértve az életpontot, éhséget, stresszt, reputációt, pénzt és tárgyakat (kenyér, vodka).
test_game.gd	A játék fő logikáját, a Family és Player objektumok integrációját, a UI komponensek betöltését, a jelenetváltás működését, valamint a Boss események és üzenetek megjelenítését ellenőrzi.

Linux alatt az alábbi paranccsal ellenőrizhető:

```shell
strace -p $game_pid -e trace=openat,rename,renameat,write
```

Példa a prototípusból, amikor a játék elmenti a settings.ini fájlt:

```strace
openat(AT_FDCWD, "/home/laptopgamer/.local/share/godot/app_userdata/SzakGyak/settings.ini.tmp", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 31
write(31, "[volume]\n\nmaster=0\nsfx=30\nmusic="..., 139) = 139
rename("/home/laptopgamer/.local/share/godot/app_userdata/SzakGyak/settings.ini.tmp", "/home/laptopgamer/.local/share/godot/app_userdata/SzakGyak/settings.ini") = 0
```

6 Összegzés

A fejlesztési folyamat során számos új ismeretre és tapasztalatra tettem szert. Kiemelt jelentősége volt a csapatmunkának: megtanultam, hogyan lehet hatékonyan együttműködni másokkal, egységes fejlesztési alapokat lefektetni, valamint közös döntéseket hozni a projekt irányáról.

Technikai téren sokat fejlődtem a grafikus felhasználói felületek tervezésében és megvalósításában, valamint a Godot motor scene- és node-alapú architektúrájának és a GDScript szignálrendszerének mélyebb megértésében. Emellett új ismereteket szereztem a dinamikus node-kezelés, a nemzetköziesítés (i18n), illetve a verziókezelési folyamatok kapcsán is. Utóbbinál különösen hasznosnak bizonyult a git eszköztár mélyebb megismerése, például a `git stash` és a `git add -p` funkciók alkalmazása.

A projekt egyik legfontosabb tanulsága az volt számomra, hogy a fejlesztési feladatok időigényét rendszerint könnyű alábecsülni. Ez a felismerés hozzájárult ahhoz, hogy a jövőben tudatosabban és reálisabban tervezzek a feladatok időbeli ütemezésével kapcsolatban.

Hivatkozások

[1] T. Salmela, „Game development using the open-source Godot Game Engine”, 2022.

[2] J. Holfeld, „On the relevance of the Godot Engine in the indie game development industry”, *arXiv preprint arXiv:2401.01909*, 2023.