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 részfeladatokat úgy általában mindenkinek a saját szemszögéből kellene bemutatnia, kihangsúlyozva a saját részt.

3.1 Előkészületek

Ezt csak úgy példának írtam, hogy lehet szépen strukturálni a dokumentumot.

3.1.1 Telepítés

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

4 Implementáció

4.1 Mentési Rendszer

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

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

```
5
6  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.

Megjegyzés

A `load_state()` korábban `Dictionary[StringName, Variant]` típust adott vissza, de a betöltés során felmerülő típusproblémák miatt végül sima `Dictionary`-ként valósítottuk meg. Ez lehetővé tette a hibamentes betöltést, miközben a különböző névterek elkülönítése továbbra is biztosított maradt.

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:

```
/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.

4.2.1 Menürendszer részletesen

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. 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.
- `new_save_menu.{gd,tscn}`: Új mentés létrehozására szolgáló felület, ahol a játékos kiválaszthatja a mentés nevét és helyét.
- `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 menü, beleértve hang, grafika és egyéb konfigurációk módosítását.
- `warn_menu.{gd,tscn}`: Tartalmi figyelmeztetések megjelenítésére szolgáló felület, amely a játék elején jelenik meg a szocialista témájú tartalom miatt.

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. 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 1 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 2 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.

Linux alatt az alábbi paranccsal ellenőrizhető:

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

Táblázat 1: 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 2: 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.

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

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

6 Összegzés

Ide érdemes leírni a munkálatokkal kapcsolatos tapasztalatokat.

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.