Beszámoló szakmai gyakorlatról

Miskolci Egyetem

Készítette:

Dobozi Botond

Programtervező informatikus

Neptun kód: HYS4P5

e-mail cím: boti2424@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 projekt egy különleges játékterv, amely a szocialista korszak mindennapjainak nehézségeit állítja a középpontba. Nem egy klasszikus akciójátékról vagy hősöket felvonultató történetről van szó, hanem egy olyan szimulációról, amely a túlélés, a döntéshozatal és a morális dilemmák kérdését vizsgálja. A játék lényege, hogy a játékos egy átlagember szemszögéből tapasztalja meg, milyen volt helytállni egy olyan rendszerben, ahol a választási lehetőségek gyakran szűkösek, a kockázatok pedig jelentősek. A projekt célja egyszerre a szórakoztatás és az elgondolkodtatás: miközben a játékos a túlélésért küzd, folyamatosan szembesül a korszak társadalmi és pszichológiai valóságával.

A célközönséget elsősorban azok a játékosok alkotják, akik értékelik a mélyebb tartalmat és a történelmi reflexiót. A játék azoknak szól, akik szeretik, ha döntéseik hosszú távon befolyásolják a játékmenetet, és nem riadnak vissza a morális dilemmáktól sem. Mivel a szocialista éra valós problémáit dolgozza fel, a játék különösen érdekes lehet történelemkedvelőknek, stratégiai játékok rajongóinak, de bárkinek, aki kíváncsi arra, hogyan éltek és túléltek emberek egy korlátozó társadalmi rendszerben. A felnőtt közönség számára ez nemcsak játék, hanem egyfajta interaktív társadalmi tükör is.

A mechanikai szinten a játék többféle munkalehetőséget kínál, például autószerelést, bolti munkát vagy irodai feladatokat. Ezek nemcsak eltérő játékmenetet biztosítanak, hanem más-más kihívásokat is hordoznak. A játékos állapotát különféle mutatók jelzik, mint az éhség, a stressz vagy a reputáció, amelyek egyensúlyban tartása folyamatos döntési kényszert eredményez. A rendszerbe beépített véletlenszerű események – például hatósági ellenőrzések vagy betegség – tovább erősítik a kiszámíthatatlanságot, így a játékosnak mindig újra kell terveznie stratégiáját. A morális dilemmák szintén hangsúlyosak: elvállaljon-e valaki kenőpénzes munkát, lopjon-e, ha családja éhezik, vagy vállalja inkább a nélkülözést?

A projekt különlegessége, hogy a megszokott nyugati kapitalista logikától eltérő játékkörnyezetet kínál. Kevés olyan játék született eddig, amely a szocializmus hétköznapi szintű tapasztalatait hitelesen mutatná be. Ez a játék képes arra, hogy a játékosokat belehelyezze egy olyan világba, ahol a mindennapi döntések komoly következményekkel járnak, és ahol a túlélés nem csupán anyagi, hanem lelki és erkölcsi kérdés is. A hitelességet tovább erősíti a gazdasági rendszer, amelyben a pénz mellett a cserekereskedelem és a feketepiac is szerepet kap.

Összességében a „Túlélés a Szocializmusban” nem csupán egy túlélő szimuláció, hanem egy interaktív történeti és társadalmi kísérlet. A játék egyszerre ad teret a stratégiai gondolkodásnak, az empátiának és a morális önvizsgálatnak. Ez a projekt azért különleges, mert képes a múlt egy darabját élményszerűen, de gondolatébresztő módon bemutatni, és ezzel egy hiánypótló szerepet betölteni a videójátékok világában.

2. Technológia kiválasztása

A projekt megvalósításához a Godot játékmotor került kiválasztásra, amely nyílt forráskódú, platformfüggetlen és alacsony rendszerigényű megoldást kínál [3]. A motor előnye, hogy jól támogatja a 2D-s és kisebb volumenű 3D-s játékok fejlesztését, emellett a tanulási görbéje kedvező a kezdő fejlesztőcsapatok számára.

A verziókövetéshez a GitHub szolgáltatása került kiválasztásra, amely biztosítja a kollaboratív fejlesztéshez szükséges átláthatóságot, a verziók nyomon követését és az

esetleges hibák visszakereshetőségét.

A rendszer modularitásra épít, ami hosszú távon egyszerűsíti a bővítéseket és az új funkciók integrálását. A választott technológiák mindegyike hozzájárul a gyors prototípus-fejlesztéshez és az iteratív munkafolyamat támogatásához. Összességében a kiválasztott eszközök megfelelő egyensúlyt teremtenek a fejlesztési rugalmasság, a fenntarthatóság és a technikai stabilitás között, így biztosítva a projekt céljainak teljesíthetőségét.

3. A tervezés folyamata

A játék fejlesztésén összesen négyen dolgoztunk. Minden csapattag egyformán kivette a részét a projekt különböző részeinek megvalósításából. Dokumentálásban, programozásban és tesztelésben, grafikai részek elkészítésében kamatoztattuk a tudásunkat valamint új ismereteket szereztünk. Az elszántság és a rugalmas hozzáállás gördülékeny munkát eredményezett a csapatunk tagjai között.

Személy szerint a projekt tervezése során új ötletekkel valamint a dokumentálás létrejöttéhez szükséges utánajárással és ennek megvalósításával tudtam plusz segítséget adni a munkálatokhoz a fent említett feladatokon kívül.

3.1. Előkészületek

A projektünk elkészítése előtt fontos volt kineveznünk egy csapatvezetőt aki koordinálja a munkát és lehetővé teszi a a kommunikációt a gyakorlatvezetőnkkel. Ez nagy szerepet játszott abban is, hogy egyes feladatok felosztása, valamint ezek határidejének betartása probléma nélkül működjön. Ezután tisztáztuk, hogy szükséges elkészítenünk egy vízió dokumentumot amely gyökeresen leírja a játék alapjait valamint, hogy miket szeretnénk pontosan elérni. Ebből a dokumentumból az általam készített rész főbb pontjai:

Felhasználói szerepkörök és projekt hatókör:

- Felhasználói szerepkörök ezen belül "Játékos", "Admin/Fejlesztő", "Vendég" céljainak leírása
- Projekt hatókör tartalmának leírása (pl: alap státuszok és mutatók), valamint ami kívül esik a hatókörön (pl: felhőalapú mentés)

3.2. Időbeosztás és ütemezés

A projekt megvalósításához összesen nyolc hét állt rendelkezésünkre, amelyet kisebb szakaszokra osztottunk fel. Az első két hétben a dokumentáció elkészítése és a játék alapjainak lefektetése volt a fő feladat. A harmadik és negyedik hétre a grafikai elemek megtervezését és a 3D modellek létrehozását tűztük ki célul. Az ötödik és hatodik héten a programozásé volt a főszerep, amikor a játékmenet alapjait és a felhasználói felület vázát építettük fel. A hetedik hét főként a hibajavításról és a rendszer finomhangolásáról szólt. A nyolcadik hétre a véglegesítettük az eddig elkészült részeket.

A feladatokat úgy osztottuk be, hogy minden szakaszban mindenki aktív szerepet kapjon, de mindig volt egy felelős, aki az adott rész haladását koordinálta. Bár előfordult, hogy egy-egy rész feladatai átnyúltak a következő hétre, a rugalmas hozzáállásnak köszönhetően sikerült tartanunk az ütemtervet.

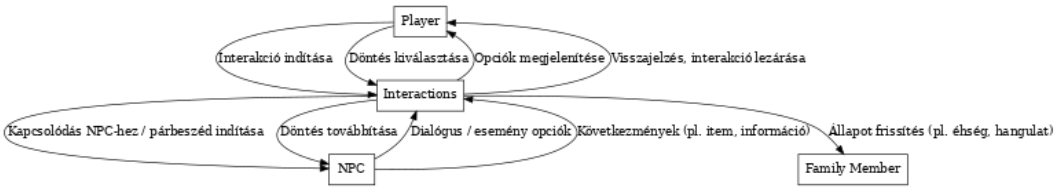
1. táblázat. Időbeosztás és ütemezés a projektre vonatkozóan

Hét	Tevékenység	Eredmény	Felelős(ök)
1–2.	Dokumentáció	Vízió dokumentum, követelmények tisztázása	Teljes csapat
3–4.	Grafikai tervezés és modellezés	2D háttérgrafikák, alap 3D modellek	Teljes csapat
5–6.	Programozás	Alap játékmenet és UI váz kialakítása	Teljes csapat
7.	Tesztelés, hibajavítás	Főbb hibák javítása, játékmenet stabilizálása	Teljes csapat
8.	Véglegesítés	Projekt részleges lezárása	Teljes csapat

3.3. Dokumentáció

A projekt során segítettem a dokumentáció elkészítését. A kezdeti lépések között szerepelt a projekt felépítésének megtervezése, a fejlesztési ütemezés rögzítése, valamint a tájékoztató jellegű dokumentumok elkészítése és beépítése a projekttervbe. A vízió dokumentum tartalmi bontását közösen végeztük el. Az SRS dokumentumon belül én állítottam össze a „Megbízhatóság”, „Teljesítmény”, „Támogatottság” és „Tervezési korlátozások” részeket.

Az analízis modell elkészítésében is közreműködtem: részt vettem az osztálydiagramok, a különféle alrendszerek és a menürendszer felépítésének kidolgozásában. Például az "Interakciók és Karakter" Alrendszeren belül az "Interakciók alrendszer" felépítésének ábrázolásával járultam hozzá a munkálatokhoz. Az analízis modell elkészítése során diagramok jelentős részét a Visual Paradigm online szerkesztő segítségével hoztunk létre [5].



1. ábra. Interakciók alrendszer

Emellett segítettem a grafikai elemek – például textúrák – hozzáadásában, illetve az általam elkészült részekben felmerülő hibák kijavításában. A dokumentációban a saját elemeim mindig naprakészek voltak a felmerülő változásokat követve.

4. Implementáció

A játék megvalósítása a Godot Engine keretrendszerben történt, amely jól támogatta a moduláris felépítést és a gyors prototípus-készítést. A projektben a jelenetek (scenes), a szkriptek és a grafikai elemek külön fájlokban kerültek kialakításra, így a csapattagok párhuzamosan tudtak dolgozni a saját moduljaikon. A verziókezelés biztosította a stabil integrációt, miközben a célunk egy játszható és stabil prototípus

létrehozása volt, amely alapként szolgált a további fejlesztésekhez. A játékmechanikák moduláris kialakítása lehetővé tette az egyszerű bővíthetőséget és karbantarthatóságot [2].

4.1. Felhasználói felület és HUD

A játék során kiemelt szerepet kapott a felhasználói felület (UI) és a HUD (Head-Up Display) kialakítása. A cél az volt, hogy a játékos egyszerre kapjon visszajelzést a főbb állapotairól (éhség, stressz, alkoholszint, pénz), miközben az interfész ne terhelje túl a képernyőt. A HUD elemei külön jelenetként lettek létrehozva, és a fő játék jelenethez kapcsolódtak, ezáltal könnyen bővíthetővé váltak. A menük és visszajelző elemek egységes színvilágot kaptak, hogy illeszkedjenek a játék hangulatához.

A megoldás előnye, hogy a HUD bármikor frissíthető a játékos aktuális állapotának megfelelően, és a játékmenet közben is reagál az eseményekre, például amikor a játékos pénzt keres vagy elveszít, illetve ha az éhség szintje megnő.

4.2. Játékmechanikák implementálása

A játék különböző mechanikáit modulárisan valósítottuk meg, így könnyen bővíthetők és karbantarthatók maradtak. Egy ilyen mechanika például az inventory rendszer, amely a játékos által gyűjtött tárgyakat és azok mennyiségét kezeli.

Az inventory megvalósítása lehetővé tette, hogy a játékos különféle tárgyakat gyűjtsön, kezeljen és felhasználjon a játékmenet során. A rendszer központi eleme egy szkript, amely egy szótárban (`Dictionary`) tartja nyilván az összes megszerzett tárgyat és azok mennyiségét. Ez biztosítja, hogy az inventory rugalmasan bővíthető legyen új tárgytípusokkal anélkül, hogy a teljes kódot át kellene alakítani.

Az alábbi kódrészlet mutatja az inventory kezelésének fő funkcióit:

```
extends Node

var items : Dictionary = {}

func add_item(name: String, amount: int = 1):
        if not items.has(name):
                items[name] = 0
        items[name] += amount

func remove_item(name: String, amount: int = 1):
        if items.has(name):
                items[name] = max(0, items[name] - amount)
                if items[name] == 0:
                        items.erase(name)

func has_item(name: String, amount: int = 1) -> bool:
        return items.has(name) and items[name] >= amount

func list_items() -> Dictionary:
        return items
```

A szkript négy fő funkciót lát el:

- `add_item`: új tárgy hozzáadása az inventory-hoz, vagy a meglévő mennyiség növelése,
- `remove_item`: tárgy mennyiségének csökkentése, illetve törlése, ha eléri a nullát,
- `has_item`: ellenőrzi, hogy a játékos rendelkezik-e egy adott tárgyból a kívánt mennyiséggel,
- `list_items`: visszaadja az inventory teljes tartalmát egy szótár formájában.

A szkript GDScript nyelven íródott, amely a Godot saját szkriptelési nyelve [1]. Ez a megoldás modulárisan illeszthető a játék más részeihez, például a HUD-hoz (tárgyak megjelenítése) vagy a játékmenet logikájához (pl. kenyér fogyasztásával csökken az éhség). Az inventory rendszer egyszerre egyszerű és hatékony, jól bővíthető a későbbi fejlesztések során.

5. Tesztelés

A projekt során a *Godot GUT* (Godot Unit Testing) [4] keretrendszert alkalmaztam az egységtesztek és integrációs tesztek elkészítéséhez. A hangsúlyt azokra a kulcsfontosságú modulokra helyeztem, amelyek a játékmenet alapját képezik: az `Inventory`, az `EventManager` és a `Stats` szkriptekre.

- **Inventory rendszer:** Az inventory működését úgy teszteltem, hogy ellenőriztem a tárgyak hozzáadását, eltávolítását, valamint a készlet ellenőrzését.

```
const Inventory = preload("res://scripts/inventory.gd")

func test_add_and_remove_item():
    var inv = Inventory.new()
    inv.add_item("bread", 2)
    assert_eq(inv.has_item("bread", 2), true)

    inv.remove_item("bread", 1)
    assert_eq(inv.has_item("bread", 2), false)
    assert_eq(inv.has_item("bread", 1), true)

func test_list_items():
    var inv = Inventory.new()
    inv.add_item("vodka", 3)
    var items = inv.list_items()
    assert_eq(items["vodka"], 3)
```

- **EventManager:** Az események kezelésénél az események véletlenszerű kiválasztását és a döntések következményeinek alkalmazását validáltam.

```
const EventManager = preload("res://scripts/event_manager.gd")
```

```
func test_trigger_event_signal():
    var em = EventManager.new()
    var called = false
    em.connect("event_triggered", func(event): called = true)
    em.trigger_random_event()
    assert_eq(called, true)

func test_apply_consequences():
    var em = EventManager.new()
    var choice = {"consequences": {"hunger": +2}}
    var stats = preload("res://scripts/stats.gd").new()
    get_tree().root.add_child(stats)
    stats.name = "Stats"
    em.apply_consequences(choice)
    assert_eq(stats.hunger, 2)
    stats.queue_free()
```

- **Stats rendszer:** Az attribútumok növelését, csökkentését és a teljes állapot lekérését teszteltem.

```
const Stats = preload("res://scripts/stats.gd")

func test_increase_and_decrease():
    var stats = Stats.new()
    stats.increase("hunger", 5)
    assert_eq(stats.hunger, 5)

    stats.decrease("hunger", 3)
    assert_eq(stats.hunger, 2)

func test_get_overall_status():
    var stats = Stats.new()
    var status = stats.get_overall_status()
    assert_eq(status.has("money"), true)
    assert_eq(status["reputation"], 50)
```

A tesztek célja az volt, hogy biztosítsák a játék fő komponenseinek stabil működését és az adatok konzisztens kezelését. Az `Inventory` tesztek ellenőrizték a tárgykezelést, az `EventManager` tesztek a döntési logika helyes futását, míg a `Stats` tesztek a karakterállapotok megbízható változását. A tesztek során a verziókövetést a GitHub repository-n keresztül kezeltük, ami lehetővé tette a hibák gyors visszakövetését.

6. Összegzés

A projekt során átfogó tapasztalatot szereztem egy játék fejlesztési folyamatáról a kezdeti tervezéstől az implementációig. A csapatmunka során részt vettem a feladatok megosztásában, a dokumentáció elkészítésében, valamint a projekt ütemezésének és

időbeosztásának megtervezésében. Fontos szerepet játszott az együttműködés és a közös ötletelés, amelyek elősegítették a gördülékeny előrehaladást.

Az implementáció során különböző technológiákat alkalmaztunk, elsősorban a Godot Engine segítségével. Megvalósítottuk a játékmenet fő elemeit, mint például az inventory rendszert, az eseménykezelőt és a statisztikákat nyilvántartó modult. Ezek moduláris felépítése lehetővé tette a kód egyszerű bővíthetőségét és átláthatóságát.

A fejlesztés mellett a dokumentáció is kiemelt figyelmet kapott. Kidolgoztam több tartalmi részt, köztük a játéktörténet, játékmenet és játéklogika leírását, valamint részt vettem az analízis modell és a diagramok elkészítésében. Ezzel nemcsak a projekt átláthatóságát biztosítottam, hanem a csapat számára is könnyebb lett követni a fejlesztési folyamatot.

A tesztelés fázisában a Godot GUT keretrendszerrel ellenőriztem az egyes modulok helyes működését. Az inventory, az EventManager és a Stats rendszerekhez írt egységtesztekkel igazoltam, hogy a logikai műveletek és az állapotváltozások megfelelően működnek. Az automatizált tesztek nagyban hozzájárultak a hibák gyors felismeréséhez és a kód stabilitásának megőrzéséhez.

A projekt során számos új ismeretet szereztem. Megtanultam, hogyan kell moduláris és bővíthető kódot írni Godot-ban, hogyan építhető fel egy játékmechanika több kisebb rendszer összehangolásával, valamint hogyan kell egységes és naprakész dokumentációt készíteni. Emellett tapasztalatot szereztem az ütemezés, a feladatok elosztása és a csapatmunka terén, ami a jövőbeli fejlesztésekben is hasznos alapot jelent. A projekt során szerzett tapasztalatok megmutatták a moduláris fejlesztés és a csapatmunka fontosságát.

Hivatkozások

[1] Godot Engine Community. Gdscript reference. https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html, 2023.

[2] Godot Engine Community. Godot engine: Best practices for modular game development. https://docs.godotengine.org/en/stable/tutorials/scripting/modular_games.html, 2023.

[3] Godot Engine Community. Godot engine documentation. https://docs.godotengine.org/, 2023.

[4] Godot Engine Community. Godot gut: Godot unit testing framework. https://github.com/bitwes/Gut, 2025.

[5] Visual Paradigm International. Visual paradigm online. https://online.visual-paradigm.com/, 2023.