Szakdolgozat

Miskolci Egyetem

Kétdimenziós játék fejlesztése Pygame keretrendszerrel

Készítette:

Szendrei Gábor

Programtervező informatikus

Témavezető:

Dr. Vadon Viktória

Miskolc, 2024

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

Szakdolgozat Feladat

Szendrei Gábor (V9ZK10) BSc programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: Játékprogramozás

A szakdolgozat címe: Kétdimenziós játék fejlesztése Pygame keretrendszerrel

A feladat részletezése:

A szakdolgozat célja egy Pygame-en alapuló kétdimenziós akció- és kalandjáték megtervezése és fejlesztése, a grafikától az implementációig. A dolgozat bemutatja az implementációhoz használt technológiákat, különös tekintettel a Python nyelvre és Pygame könyvtárra, miért alkalmasak kalandjáték fejlesztésére, és összehasonlítja őket egyéb alternatívákkal. A dolgozat bemutatja az akció- és kalandjátékok általános jellemzőit, és ezt vesszük inspirációnak a fejlesztett játék megtervezéséhez. Mint kalandjátékokra jellemző, a fejlesztett játékban a cselekménysort egy küldetés rendszer adja, ehhez megvalósításra kerülnek különböző interakciók NPC-kkel. Az alapvető játékfunkciókon (karakter irányítása, pályával való interakció, támadás) felül a még jobb játékélmény érdekében megvalósításra kerülnek egyéb játékfunkciók is, mint a pálya animációja és hangeffektek, illetve egy pillanatállj funkció. Továbbá megvalósításra kerül egy bejelentkezési rendszer, mely lehetőséget ad az előrehaladás mentésére, illetve a legjobb egyéni eredmények mentésére egy online adatbázisban és versenyre más játékosok rekordjaival.

Témavezető: Dr. Vadon Viktória, adjunktus

A feladat kiadásának ideje: 2023. szeptember 21.

.....
szakfelelős

Eredetiségi Nyilatkozat

Alulírott **Szendrei Gábor**; Neptun-kód: `V9ZK10` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Szakdolgozat Címe* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ... év ..... hó ..... nap

.....
Hallgató

1. A szakdolgozat feladat módosítása
   - szükséges (módosítás külön lapon)
   - nem szükséges

.....	.....
dátum	témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás):	konzulens (dátum, aláírás):
.....	.....
.....	.....
.....	.....

3. A szakdolgozat beadható:

.....	.....
dátum	témavezető(k)

4. A szakdolgozat ..... szövegoldalt
   ..... program protokollt (listát, felhasználói leírást)
   ..... elektronikus adathordozót (részletezve)
   .....
   ..... egyéb mellékletet (részletezve)
   .....
tartalmaz.

.....	.....
dátum	témavezető(k)

5. A szakdolgozat bírálatra
   - bocsátható
   - nem bocsátható

A bíráló neve: .....

.....	.....
dátum	szakfelelős

6. A szakdolgozat osztályzata

   a témavezető javaslata: .....
   a bíráló javaslata: .....
   a szakdolgozat végleges eredménye: .....

Miskolc, ...
a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. Bevezetés **1**

2. Játékfejlesztés és a Pygame **2**
2.1. Bevezetés a játékfejlesztés világába 2
2.1.1. Mit jelent játékot fejleszteni? 2
2.1.2. A játékfejlesztés kihívásai és lehetőségei 2
2.2. Játékfejlesztő könyvtárak és eszközök 3
2.2.1. Unity: a 3D játékfejlesztés platformja 3
2.2.2. C#: Kiemelt szerepű programozási nyelv a játékfejlesztők körében 3
2.2.3. Java: népszerű választás játékfejlesztők számára 4
2.2.4. C++: A régebbi játékok elengedhetetlen nyelve 4
2.3. Python a játékfejlesztésben 5
2.3.1. Miért jó a Python? 5
2.3.2. A Python és a játékfejlesztés 5
2.4. Pygame 6
2.4.1. Pygame-mel készített sikeres játékok 6
2.4.2. Pygame lehetőségei 6
2.4.3. Prototípusok és koncepciók 7
2.4.4. További tudnivalók a Pygame-ről 7
2.4.5. Összegzés 7
2.5. Pygame és Ren'py összehasonlítása 7
2.5.1. Felhasználási terület 7
2.5.2. Célcsoport 8
2.5.3. Felhasználói felület 8
2.5.4. Felépítmény 8
2.5.5. Funkciók 8
2.5.6. Összegzés 9

3. Követelmények a játékkal szemben **10**
3.1. Felhasználói élmény kialakítása 10
3.2. Grafika 10
3.3. Menürendszer 10
3.3.1. Felhasználói fiók 11
3.3.2. Főmenü funkciói 11
3.3.3. Játékon belüli menürendszer 12
3.4. Felhasználói felület és a játékos cselekvési lehetőségei 12
3.4.1. Felhasználói felület 12
3.4.2. Életpontok és energia 13

3.4.3. Fegyverválasztás . . . 13
3.4.4. Harcolás szörnyekkel . . . 13
3.4.5. Küldetések felvétele . . . 14
3.4.6. Bájitalok vásárlása és használata . . . 14
3.4.7. Statisztikák növelése tapasztalati pontokból . . . 14

4. Megvalósítás **15**
4.1. Grafikák elkészítése . . . 15
4.1.1. Grafikus szerkesztő . . . 15
4.1.2. Rajzok elkészítése . . . 16
4.2. Pálya megtervezése . . . 17
4.2.1. Tiled . . . 17
4.2.2. Tervezés . . . 17
4.3. Felhasználói felület és beállítások kezelése . . . 18
4.3.1. Beállítások . . . 18
4.3.2. Felhasználói fiók . . . 18
4.3.3. Főmenü . . . 19
4.3.4. Új játék létrehozása . . . 19
4.3.5. Játékmenet közbeni menü . . . 20
4.3.6. Mentések betöltése . . . 21
4.4. Játékmenet kezelése . . . 21
4.4.1. Main metódus . . . 21
4.4.2. Játékon belüli felületek . . . 22
4.4.3. Játékállapot mentése . . . 22
4.4.4. Pálya kezelése . . . 23
4.4.5. Világ és barlangok . . . 23
4.4.6. Entitások . . . 25
4.4.7. Játékos karaktere . . . 27
4.4.8. Fegyverek . . . 27
4.4.9. Ellenségek, szörnyek . . . 28
4.4.10. Zsákmányolás . . . 29
4.4.11. Nem a játékos álltal vezértelt karakterek . . . 30
4.4.12. Küldetést adó karakterek . . . 30
4.4.13. Kereskedő karakter . . . 31
4.4.14. Táska rendszer . . . 32
4.5. Adatbázis használata a programban . . . 32

5. Továbbfejlesztési lehetőségek **33**
5.1. Táska rendszer . . . 33
5.2. Pálya generálás . . . 33
5.3. Ellenségek . . . 33
5.4. Felhasználói fiókok biztonságosabbá tétele . . . 34

6. Összefoglalás **35**

7. Summary **36**

Források **37**

1. fejezet

Bevezetés

A számítógépes játékok már régóta a szórakozás egyik legnépszerűbb formáját képviselik. Az idők során folyamatos fejlődésen mentek keresztül, és ma már számtalan lehetőséget kínálnak a játékosoknak.

A számítógépes játékok lehetővé teszik számunkra, hogy elmerüljünk olyan világokban, amelyek különböző fantáziavilágokban játszódnak, izgalmas cselekmények részesei lehetünk, és akár versenyezhetünk játékosokkal a világ minden tájáról. Az interaktivitás a játékok egyik fő jellemzője, hiszen a játékosok döntéseket hozhatnak, irányíthatják a karaktereket és befolyásolhatják a játék alakulását, így személyes élményeket és kalandokat élhetnek át.

A számítógépes játékokban a grafika és a hang is kulcsfontosságú szerepet játszik. A modern grafikus motorok és a kiváló minőségű hangeffektek valósághű és lenyűgöző világokat teremtenek, amelyekbe a játékosok könnyedén belefeledkezhetnek. A 3D-s grafika és a virtuális valóság (VR) technológia pedig még inkább fokozza a valóságérzetet, lehetővé téve, hogy teljes mértékben elmerüljünk a játék világában.

A számítógépes játékoknak számos műfaja létezik, így mindenki megtalálhatja a saját ízlésének megfelelő játékot. Akciójátékok, stratégiai játékok, szerepjátékok, sportjátékok, horror játékok és még sok más közül választhatunk, mindegyik más-más kihívásokkal és élményekkel gazdagít.

Emellett a számítógépes játékoknak társadalmi szerepük is van. A többjátékos módban lehetőség van csapatban játszani, vagy akár versenyezni más játékosokkal online. Ez segít kapcsolatokat építeni és barátokat szerezni.

Az eSport egyre népszerűbbé válik, és hatalmas növekedési potenciállal rendelkezik. Versenyek és ligák alakulnak ki, ahol a legjobb játékosok hatalmas pénzdíjakért versenyeznek, és mindezt élvezhetik a nézők is.

Összességében a számítógépes játékok kiváló lehetőséget nyújtanak a szórakozásra, és a technológia folyamatos fejlődésének köszönhetően még évekig fontos szerepet fognak játszani a szórakoztatóiparban.

A szakdolgozatom célja egy 2-dimenziós akció-kaland játék fejlesztése, melyben a játékosok teljesítsenek különféle izgalmas küldetéseket, miközben felfedezik a szigetet és barlangjait, és kihívásokkal teli kalandokban vesznek részt.

Dolgozatomban részletes betekintést nyújtok a Python programozási nyelv és a pygame könyvtár használatába. Emellett összehasonlító elemzést végezek a pygame és a Ren'py könyvtárak között, és bemutatom a játék implementációjának részleteit is.

2. fejezet

Játékfejlesztés és a Pygame

2.1. Bevezetés a játékfejlesztés világába

A játékfejlesztés a modern szórakoztatóipar egyik legszerteágazóbb és legizgalmasabb területe, amely számtalan lehetőséget rejt magában a kreativitás kibontakoztatására és az élmények megteremtésére. Ennek a fejezetnek az elején bemutatom, hogy mit jelent játékot fejleszteni, milyen kihívásokkal jár, és milyen lehetőségeket kínál.

2.1.1. Mit jelent játékot fejleszteni?

Játékot fejleszteni egy olyan folyamat, amely során valamilyen virtuális alkalmazást tervezünk, készítünk és tesztelünk. Ezek a játékok szórakoztatnak, kihívások elé állítanak, vagy éppen történeteket mesélnek el a játékosoknak. A játékfejlesztés során számos különböző területet érintünk, mint például a grafika, a hang, a programozás, a játéktervezés és a narratíva.

A játékfejlesztés során a következő elemeket kell figyelembe venni:

- Játéktervezés: A játékmechanizmusok, pályatervezés, karakterek és történet kidolgozása.
- Grafika és dizájn: A játékvilág megtervezése, karakterek és tájak kinézetének megalkotása.
- Hang és zene: A játékhangulat meghatározása zenei és hanghatásokkal.
- Programozás: A játék mechanizmusainak és logikájának implementálása.

2.1.2. A játékfejlesztés kihívásai és lehetőségei

A játékfejlesztés izgalmas, de komplex folyamat, amely számos kihívást rejt magában.

- **Technikai kihívások:** A játékfejlesztéshez fejlett szoftveres és hardveres ismeretekre van szükség. A játékmotorok, programozási nyelvek és grafikai eszközök használata összetett feladatokkal jár.
- **Projektmenedzsment:** A játékfejlesztés projektek hosszú és komplex folyamatok, amelyek határidőket és költségvetéseket igényelnek. A megfelelő projektmenedzsment kritikus fontosságú.

- **Felhasználói élmény:** A játékosok elégedettségének és élvezetének biztosítása kiemelten fontos. A játéktervezés és felhasználói felület optimalizálása elengedhetetlen.

Ugyanakkor a játékfejlesztésben óriási lehetőségek is rejlenek:

- **Kreativitás:** A játékfejlesztés lehetőséget ad a kreativitás megnyilvánulására, ahol csak a képzelet szab határt. A jó játékok egyediséget és kreativitást követelnek meg a játéktervezőktől. Hiszen az új és izgalmas játékmechanizmusok kitalálása kulcsfontosságú.
- **Közösség és verseny:** A játékfejlesztők részt vehetnek aktív közösségekben, és akár versenyeken is, ahol megmutathatják tehetségüket és fejlődésüket. Ilyen verseny például a *The Game Development World Championship* [1].
- **Szórakoztatás:** A jó játékok milliók számára jelentenek kikapcsolódást és szórakozást, és hűséges rajongótábort hozhatnak létre.

2.2. Játékfejlesztő könyvtárak és eszközök

A játékfejlesztéshez elérhető könyvtárak és eszközök rendkívül fontosak a fejlesztők számára, hiszen segítenek a játékok fejlesztésében és optimalizálásában. Ebben a részben áttekintünk néhány közismert könyvtárat és eszközt, amelyeket játékfejlesztéshez használnak.

2.2.1. Unity: a 3D játékfejlesztés platformja

A Unity [2, 3] a játékfejlesztők körében rendkívül népszerű, mivel egyszerűen kezelhető és széleskörű lehetőségeket kínál a játékok létrehozásához. A platform komplex fejlesztői eszközökkel rendelkezik, például szkriptelési lehetőségekkel és beépített folyamatkezelővel, amelyek megkönnyítik a játékfejlesztést.

A Unity támogatja a 2D és 3D játékok készítését egyaránt, így a fejlesztők szabadon választhatják meg a stílust és a műfajt. A platform lehetővé teszi a cross-platform fejlesztést, amely azt jelenti, hogy egyetlen projektből készíthetünk játékokat különböző platformokra.

A Unity erős grafikai motorokkal rendelkezik, amelyek lehetővé teszik a gyönyörű és részletes grafikák létrehozását. Emellett fizikai motorjai valósághű mozgást és ütközéseket biztosítanak, ami a játékélményt még valóságosabbá teszi.

A folyamatos támogatás és a nagy közösség miatt a Unity egy kiváló választás a játékfejlesztők számára, akik minőségi játékokat szeretnének létrehozni a különböző platformokon. A Unity segítségével a fejlesztők könnyen hozzáférhetnek az új funkciókhoz és frissítésekhez, hogy folyamatosan fejleszthessék játékaikat.

2.2.2. C#: Kiemelt szerepű programozási nyelv a játékfejlesztők körében

A C# [4] egy modern, objektumorientált programozási nyelv, melyet széles körben alkalmaznak a játékfejlesztés területén, különösen a Unity játékmotorral[5]. A Unity

egyike a legnépszerűbb játékfejlesztő platformoknak, és C#-t használ a játéklogika és szkriptelés megvalósításához, így lehetővé téve a fejlesztők számára a cross-platform játékok készítését.

A C# programokat általában a Visual Studio[6] fejlesztői környezetben írják, amely számos hasznos eszközt és szolgáltatást kínál a fejlesztőknek, például hibakeresőt, kódszerkesztőt és verziókezelőt. A Visual Studio lehetővé teszi a fejlesztők számára a kódolás, tesztelés és hibakeresés egyszerű és hatékony módját.

A Unity és C# együttes használata lehetővé teszi a fejlesztők számára a könnyű és hatékony játékfejlesztést számos platformra, mint például számítógépek, mobil eszközök és konzolok. Ez a kombináció erős grafikai motorokkal, fizikai motorokkal és egyéb eszközökkel is rendelkezik, amelyek segítik a játékélmény kialakítását és optimalizálását.

Mivel a C# egy népszerű és jól támogatott nyelv a játékfejlesztésben, a fejlesztők könnyen hozzáférhetnek különböző fejlesztői eszközökhöz és könyvtárakhoz, amelyek elősegítik a játékfejlesztést és a játéktervezést.

2.2.3. Java: népszerű választás játékfejlesztők számára

A Java [7, 8] egy platformfüggetlen, objektumorientált programozási nyelv, amelyet a játékfejlesztésben is széles körben alkalmaznak, különösen az Android platformon. A Java játékok fejlesztéséhez különféle fejlesztői eszközök és könyvtárak állnak rendelkezésre.

Például, a Java játékok fejlesztéséhez számos integrált fejlesztői környezet (IDE) érhető el, mint például az Android Studio vagy a Eclipse, amelyek segítenek a játékok tervezésében és kódolásában. Ezek az IDE-k számos hasznos eszközt és szolgáltatást kínálnak a fejlesztőknek, például hibakeresőt és kódszerkesztőt.

A Java játékok grafikai részének fejlesztéséhez számos grafikai könyvtár is rendelkezésre áll, például a LibGDX [9] vagy a JavaFX [10]. Ezek a könyvtárak lehetővé teszik a játékgrafikák létrehozását, animációk kezelését és a felhasználói felület kialakítását.

Emellett a Java egy erős közösséggel rendelkezik, ami azt jelenti, hogy a fejlesztők könnyen hozzáférhetnek különböző fejlesztői eszközökhöz és könyvtárakhoz, amelyek segítik a játékfejlesztést. Példaként említhetők a játékfejlesztéshez használt függvénykönyvtárak, például a LWJGL (Lightweight Java Game Library) [11], amely lehetővé teszi a háromdimenziós játékok fejlesztését. Emellett fórumok és közösségi oldalak is rendelkezésre állnak a fejlesztők számára, ahol tapasztalatokat cserélhetnek és segítséget kérhetnek egymástól, a leghíresebb ilyen a *Java Prgoramming Forums* [12]. Ezen a weboldalon a fejlesztők különböző témákban kérhetnek segítséget, vagy éppen segíthetnek másoknak a problémáik megoldásában. Remek leírások, példák és cikkek is találhatóak rajta.

2.2.4. C++: A régebbi játékok elengedhetetlen nyelve

A C++ [13, 14] egy erőteljes és hatékony programozási nyelv, amely gyakran előfordul a játékfejlesztés világában, különösen a nagy teljesítményű játékok és konzolplatformok esetében. A C++ nyelv lehetővé teszi a fejlesztők számára, hogy közvetlenül a hardverre programozzanak, ami rendkívül nagy szabadságot és teljesítményt nyújt.

A játékfejlesztők körében a C++ kiemelkedően kedvelt nyelv, mivel lehetőséget nyújt a nagy teljesítményű játékok létrehozására és a hardverrel való közvetlen kapcso-

lat kialakítására. Számos játékfejlesztő könyvtár és motor támogatja a C++ nyelvet, amelyek segítik a játékfejlesztőket a projektjeik gyorsabb és hatékonyabb megvalósításában.

2.3. Python a játékfejlesztésben

A Python [16] egy kiválóan használható programozási nyelv a játékfejlesztéshez, és sok előnnyel rendelkezik a fejlesztők és a játéktervezők számára. Ebben a fejezetben kifejtem, miért érdemes Pythonnal dolgozni játékok tervezésekor, és milyen alapvető tulajdonságok és lehetőségek teszik ezt a nyelvet vonzóvá a játékfejlesztés világában.

2.3.1. Miért jó a Python?

A Python népszerűségének [17] több oka van:

- **Olvashatóság és egyszerűség:** Python kódot írni könnyű és gyors. A Python nyelv szintaxisa rendkívül olvasható és hasonlít az angol nyelvre, ami megkönnyíti a kód értelmezését. A könnyű olvashatóság csökkentheti a fejlesztési időt és a hibák számát egyaránt.
- **Gyors fejlesztés:** Lehetővé teszi a gyors prototípusok létrehozását. A gyors prototípusok segítenek a játék ötleteinek gyors validálásában és tesztelésében, mielőtt hosszú fejlesztési ciklusokba kezdünk.
- **Széleskörű támogatás és közösség:** Rendelkezik egy nagy és elkötelezett fejlesztői közösséggel, ami számos kiegészítő könyvtárat és eszközt kínál a játékfejlesztőknek. Ez a közösség folyamatosan fejleszti és frissíti a nyelvet és az eszközöket.

2.3.2. A Python és a játékfejlesztés

A Python programozási nyelvet egyre gyakrabban alkalmazzák a játékfejlesztés területén. [15] Bár eredetileg nem a legnépszerűbb választás volt ebben a szektorban, az utóbbi években számos előnye miatt kezdett teret hódítani.

A Python játékfejlesztésre való áttérést elősegíti az egyszerűsége és olvashatósága, amely lehetővé teszi a fejlesztők számára, hogy a játékmechanizmusokra és a játékélményre összpontosítsanak, anélkül hogy túlzottan mélyre kellene merülniük a technikai részletekbe.

Ezenkívül a Python platformfüggetlen, így a fejlesztők könnyedén exportálhatják játékaikat különböző rendszerekre, például Windows, macOS vagy Linux alá, ami tovább növeli a nyelv vonzerejét a játékfejlesztők számára.

Mivel a Python a játékfejlesztés területén egyre népszerűbbé válik, egyre több játék fejlesztése zajlik ezen a platformon, és továbbra is új lehetőségek és fejlesztői eszközök válnak elérhetővé a játékfejlesztők számára. Ezek közül az egyik legnépszerűbb a Pygame.

2.4. Pygame

A Pygame [18] egy nyílt forráskódú programozási platform és könyvtár, mely lehetővé teszi játékok és multimédiás alkalmazások fejlesztését a Python programozási nyelv segítségével. Különösen népszerű a játékfejlesztők körében, mivel könnyen használható és rendkívül rugalmas. A Pygame rétegekben épül fel, amelyek lehetővé teszik a felhasználók számára, hogy kezeljék az ablakokat, az eseményeket, a grafikát és a hangot.

2.4.1. Pygame-mel készített sikeres játékok

A Pygame lehetővé teszi a fejlesztők számára, hogy kreatív és sokoldalú játékokat hozzanak létre. Néhány példa sikeres Pygame projektekre:

- **Cave Story [20] (2004):** Egy kalandjáték, amelyet Daisuke Amaya készített. A játék nagy sikert aratott a játékosok körében, és azóta is népszerű maradt.
- **Super Meat Boy [21] (2010):** Egy gyors tempójú platformjáték, amelyben a játékos egy húsból készült fiút irányít, aki megpróbál eljutni egy húsból készült lányhoz.
- **Braid [22] (2008):** Egy különleges platformjáték, amelyben a játékosnak időutazást kell alkalmaznia a játék különböző szintjein.

2.4.2. Pygame lehetőségei

A Pygame rendkívül sokoldalú eszközöket és lehetőségeket kínál a játékfejlesztők számára.

- **Grafikai megjelenítés:** Lehetővé teszi a grafikus elemek létrehozását és kezelését, beleértve a felületeket, háttérképeket és megjelenítési funkciókat is.
- **Hangkezelés:** A könyvtár lehetővé teszi hangfájlok lejátszását, hanghatások és zene hozzáadását a játékhoz.
- **Felhasználói interakciók:** Segítségével könnyedén kezelhetők a felhasználói inputok, például a billentyűzet és egér események.
- **Multiplatform támogatás és hordozhatóság:** Rendkívül hordozható, és szinte minden platformon és operációs rendszeren fut, beleértve Linuxot, Windowst, MacOS-t és másokat. Alkalmazható számos eszközön és operációs rendszeren, beleértve a kézi eszközöket, játékkonzolokat és a One Laptop Per Child (OLPC) [19] számítógépét is.
- **Egyszerűség:** Könnyen megtanulható és használható, és kiválóan alkalmas fiatalabb és idősebb játékfejlesztők számára egyaránt.
- **Modularitás:** Megadja a lehetőséget, hogy a különböző modulokat külön-külön inicializáljuk és használjuk, ezáltal a felhasználó személyre szabhatja a fejlesztést az igényei szerint.

2.4.3. Prototípusok és koncepciók

A Pygame lehetővé teszi a gyors prototípusok készítését és koncepciók tesztelését a valóságban. A prototípusok és koncepciók tesztelése segíthet a fejlesztőknek abban, hogy javítsák a játékaikat, mielőtt azok nagyszabású fejlesztésbe kerülnek. A Pygame-et gyakran használják új játékmechanikák, játékstílusok tesztelésére, mivel gyorsan és hatékonyan lehet vele prototípusokat készíteni.

2.4.4. További tudnivalók a Pygame-ről

- **Széleskörű közösség és források:** A Pygame hatalmas fejlesztői közösséggel rendelkezik, ami azt jelenti, hogy rengeteg dokumentáció, tutorial és fórum áll rendelkezésre a segítségnyújtáshoz és a problémamegoldáshoz. Az aktív közösség folyamatosan fejleszti és frissíti a Pygame-et, így a fejlesztők mindig naprakész forrásokhoz férhetnek hozzá.
- **Könnyen tanulható:** A Pygame olyan egyszerűen használható, hogy akár gyerekek és fiatalabb játékfejlesztők is könnyen megtanulhatják a használatát. A kezdeti lépések után a fejlesztők gyorsan építhetnek fel játékokat és alkalmazásokat a Pygame segítségével.

2.4.5. Összegzés

A Pygame és hasonló játékfejlesztő eszközök széles körű alkalmazást kínálnak a modern társadalomban. Ezek az eszközök nem csupán játékok készítésére használhatók, hanem hatékony eszközök a tanulás, a kutatás és a kreativitás terén is. A játékosított tanulás révén motiválóbbá tehetjük az oktatást. Emellett a játékfejlesztés lehetőséget ad az ötletelésre és a kreatív kifejezésre is. Bármilyen szinten is legyen valaki a játékfejlesztés terén, a Pygame és hasonló eszközök segítségével saját projekteket hozhat létre és hozzájárulhat a tudásunk bővítéséhez és a különböző területeken való alkalmazásához. A játékok nagyon hasznos eszközök, amelyeket a gyerekek és felnőttek is élveznek.

2.5. Pygame és Ren'py összehasonlítása

A Pygame és a Ren'Py [23] két olyan kiváló szoftverkönyvtár, amelyeket a játékfejlesztők és vizuális novellák készítői használnak világszerte. Bár első pillantásra talán nincs sok közös vonásuk, mindkét platform a játékok és interaktív történetek fejlesztésére szolgál, és számos hasonlóság és különbség rejlik bennük. Ebben a fejezetben részletesen megvizsgáljuk a Pygame és a Ren'Py funkcióit, lehetőségeit, valamint az alkalmazásukat különböző projektekhez. Ezzel a közvetlen összehasonlítással [24] lehetőséget teremtünk arra, hogy megtudjuk, melyik könyvtár a legalkalmasabb az adott célkitűzések és projektek megvalósítására, és milyen előnyökkel és korlátozásokkal jár az alkalmazásuk.

2.5.1. Felhasználási terület

A Pygame és a Ren'Py két különböző, de rendkívül hasznos eszköz a játékfejlesztők és vizuális novellák alkotói számára. A Pygame sokoldalúságának köszönhetően szinte

bármilyen típusú játékfejlesztésre alkalmazható, így lehetőséget biztosít akció-játékok, platformjátékok, vagy éppen puzzle-játékok létrehozására. Azonban a Ren'Py specifikus specializációja a szövegalapú vizuális novellák, interaktív történetek készítésére szorítkozik, és kiemelt figyelmet fordít a narratívára, karakterek párbeszédeire és képeire. Ezáltal mindkét eszköz egyedi felhasználási területtel rendelkezik, és a választás az alkotók célkitűzéseitől és projektjeitől függ. A Pygame sokféle játékstílus és ötlet megvalósítására alkalmas, míg a Ren'Py a történetmesélés és karakterfejlődés hangsúlyozásával ideális választás azok számára, akik szövegalapú interaktív élményeket szeretnének létrehozni.

2.5.2. Célcsoport

A Pygame általános célú keretrendszerként szolgál, ami azt jelenti, hogy szinte bármilyen típusú játék készítéséhez használható. Legyen szó akció-játékról, platformjátékról, vagy akár puzzle-játékról. A Ren'Py viszont kifejezetten a szövegalapú visual novelek (interaktív történetek) készítésére specializálódott. Itt a fő hangsúly a narratíván, karakterek párbeszédein és képeken van.

2.5.3. Felhasználói felület

Bár a Pygame keretrendszer használata viszonylag egyszerű, az alapvető programozási ismeretek elengedhetetlenek. A fejlesztőknek szükségük van jártasságra a Python programozási nyelvben és az alapvető játékfejlesztési technikákban.

A Ren'Py használata rendkívül intuitív és könnyen érthető, még azok számára is, akiknek nincsen tapasztalata a programozásban. Ennek köszönhetően a felhasználók könnyedén létrehozhatnak interaktív szöveges játékokat anélkül, hogy mély programozási ismeretekkel rendelkeznének. Tudás és tapasztalat függvényében akár Pygame-et is használhatunk a Ren'Py játékunkban, például minigame-ek készítésére.

2.5.4. Felépítmény

A Pygame egy objektumorientált keretrendszer, ahol a játékot különböző objektumokból építik fel, és a fejlesztőknek az objektumok közötti interakciókat kell kezelniük. A Ren'Py egy szöveg-alapú keretrendszer, ahol a játékot szövegből és grafikából építik fel. Az objektumok és interakciók kezelése itt kevésbé hangsúlyos, a fókusz a narratívára összpontosul.

2.5.5. Funkciók

A Pygame számos alapvető funkciót kínál, mint például a grafika, a hang és a bevitel kezelése. Fejlesztőknek nagyobb szabadságot ad azáltal, hogy saját logikát és rendszereket hozhatnak létre.

A Ren'Py speciális funkciókat kínál az interaktív történetek készítéséhez, mint például a szöveg effektek, a zene és a képkockák kezelése. A keretrendszer célja a visual novel műfaj specifikus igényeinek kielégítése.

2.5.6. Összegzés

A megfelelő keretrendszer kiválasztása segíthet abban, hogy gyorsabban és könnyebben készítsen professzionális minőségű játékokat.

A Pygame és a Ren'Py két népszerű keretrendszer a játékfejlesztéshez. A Pygame általános célú keretrendszer, amely bármilyen típusú játék készítéséhez használható. A Ren'Py pedig egy interaktív történetek készítésére specializált keretrendszer.

A két keretrendszer kiválasztásakor a következő szempontokat érdemes figyelembe venni: Játékstílus, milyen típusú játékot szeretne készíteni? Tapasztalat, mennyi programozási tapasztalattal rendelkezik?

3. fejezet

Követelmények a játékommal szemben

Játék tervezésénél fontos szem előtt tartani azokat a követelményeket, amelyeknek meg kell felelnie. A felhasználói élmény szorosan kapcsolódik ahhoz, hogy a játékosok mennyire értik meg a játék működését, és ezen élmény tervezése alapvető fontosságú.

3.1. Felhasználói élmény kialakítása

A felhasználói élmény kialakítása során kiemelt figyelmet kell fordítani az elrendezésre, az egyértelmű utasításokra. A játékosoknak világosan kell látniuk, hogy hogyan tudnak interakcióba lépni a játék világával, és ezek az elemek nagyban hozzájárulnak a felhasználói élmény minőségéhez.

Az érthető és könnyen kezelhető felhasználói felület kulcsfontosságú a játék sikerességéhez. A játékosoknak könnyen kell tudniuk kezelni a játék funkcióit és lehetőségeit, hogy maximálisan élvezhessék a játékot.

Az is fontos tényező, hogy a játék érthetősége és használhatósága ne csak a tapasztalt játékosok számára legyen megfelelő, hanem a kezdők és a kevésbé jártas személyek számára is könnyen hozzáférhető legyen. Az egyszerű és intuitív felület tervezése segíthet abban, hogy minél több játékos élvezhesse a játékot.

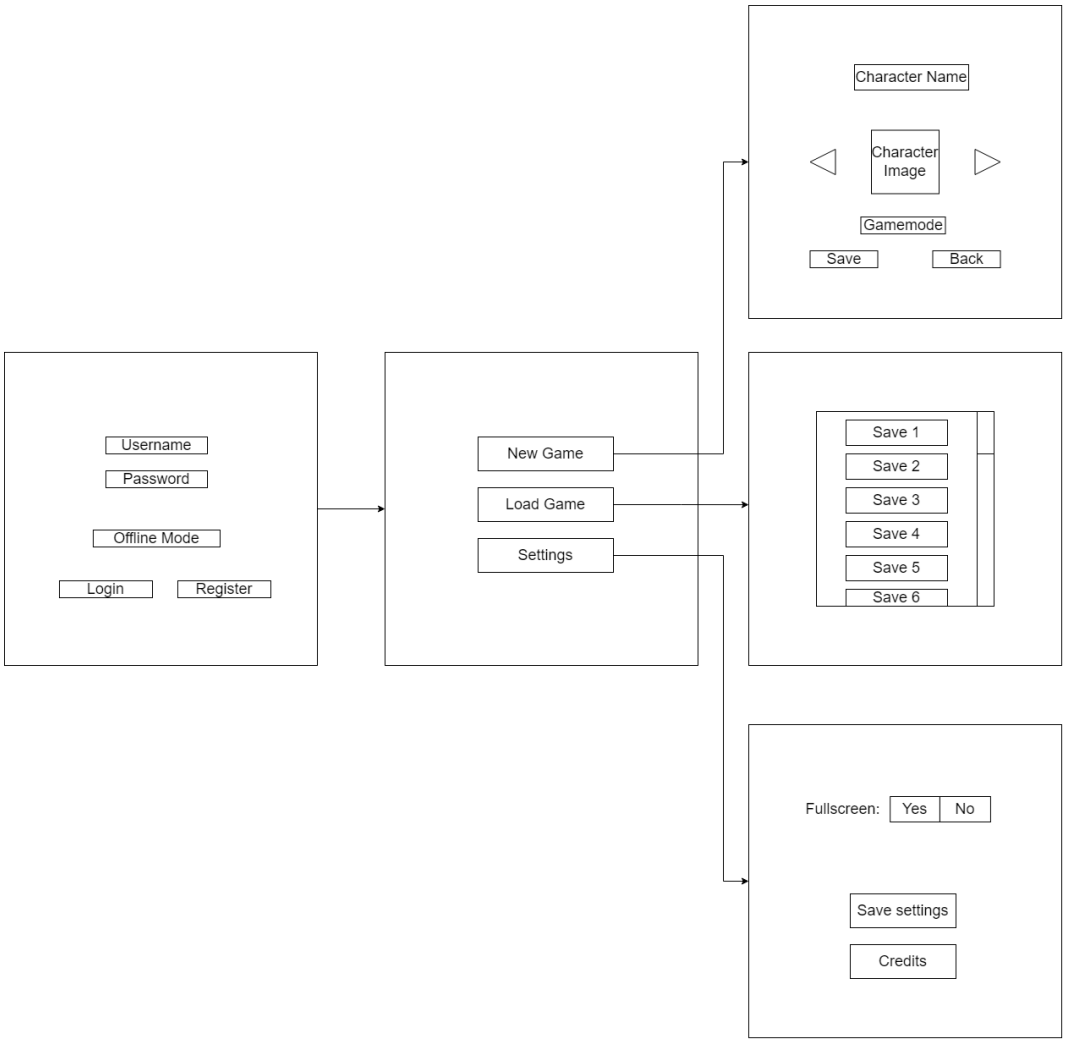
3.2. Grafika

A játékok szempontjából a grafika kulcsfontosságú elem. Egy gondosan kidolgozott vizuális megjelenés mélyebben bevonja a játékosokat a játék világába, ami növeli a játék élvezetét.

Ezért fontos számomra, hogy saját magam készítsem el a rajzokat, ezzel is egyedivé téve a játékot, illetve a fejlesztés során egyszerűbben áthidalhatóak a grafikai problémák, hiszen nem kell harmadik féllel egyeztetnem. A pálya kirajzolásánál szeretnék megvalósítani hamis 3D-s hatást, hogy a játékosok számára élethűbbnek tűnjön a játék.

3.3. Menürendszer

A menürendszernek általában a játékhoz illőnek kell lennie, hogy a játékosok ne érezzék azt, hogy egy másik játékot indítanak el. Fontos az is, hogy átlátható legyen, hogy a játékosok könnyen eligazodjanak benne, és ne kelljen sok időt tölteniük azzal, hogy megtalálják a keresett funkciót. A menük közti navigációt a 3.1 ábra szemlélteti.



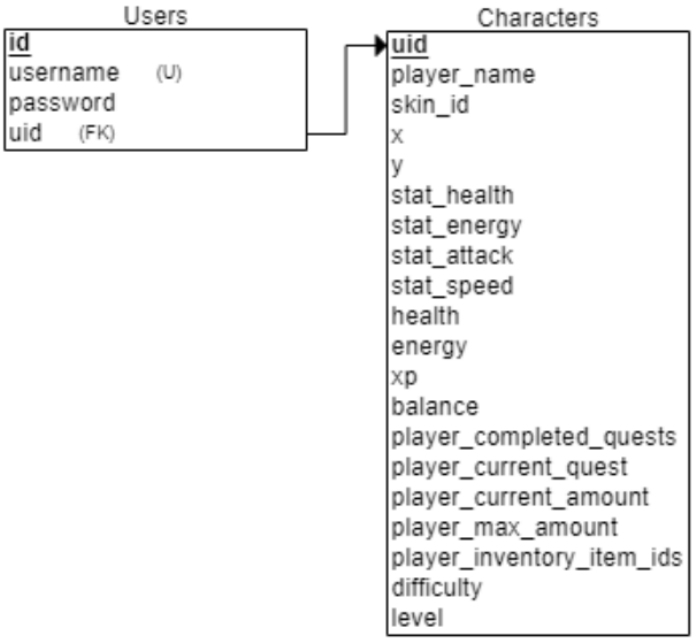
3.1. ábra: Menürendszer terve

3.3.1. Felhasználói fiók

Felhasználói fiókok létrehozása azért volt szükséges a játékomban, mivel szerettem volna létrehozni egy ranglista rendszert, ahol a játékosok tudhatták, hogy egymással versenyeznek, hogy ki hányszor vitte végig a játékot anélkül, hogy a karakterük egyszer is meghalt volna, miközben a szörnyek minden újrakezdés után erősödtek. Ehhez a rendszerhez elengedhetetlen volt, hogy meg tudjam különböztetni a játékosokat, illetve, hogy egy adatbázisban tudjam tárolni a játékosok adatait. Az adatbázis relációs modellje a 3.2 ábrán látható.

3.3.2. Főmenü funkciói

A bejelentkezést követően három menüpont válik elérhetővé a játékosok számára. Az egyik kiemelkedően fontos lehetőség a beállítások menüpont, mivel általában itt találhatók a kulcsfontosságú funkciók módosítására szolgáló lehetőségek. Az én esetemben itt érhető el a teljes képernyős mód beállítása. A beállítások menüpont alatt továbbá megtalálható egy "Credits" opció is, ami tartalmazza a készítő nevét és az elkészítés évét. A másik két menüpont a játék indításával kapcsolatos: lehetőség van új karakter létrehozására, illetve meglévő mentéseik betöltésére.



3.2. ábra: Relációs modell

3.3.3. Játékon belüli menürendszer

A legtöbb játékban kiemelkedő jelentőségű a játékon belüli pillanat állj funkció, és egy ilyenkor megjelenő belső menürendszer. Lényeges, hogy ezek könnyen elérhetőek legyenek, miközben nem zavarják a játékélményt. A játékosoknak lehetőségük van a játék közben is hozzáférni egy beállítások menühöz, amelyben a hangerőszintet is be tudják állítani. Ezen menü részét kell képeznie egy mentés funkciónak, ami a játék aktuális állapotának mentését teszi lehetővé, továbbá egy folytatás lehetőségnek, és egy kilépés opciónak, amely a játék bezárását szolgálja.

3.4. Felhasználói felület és a játékos cselekvési lehetőségei

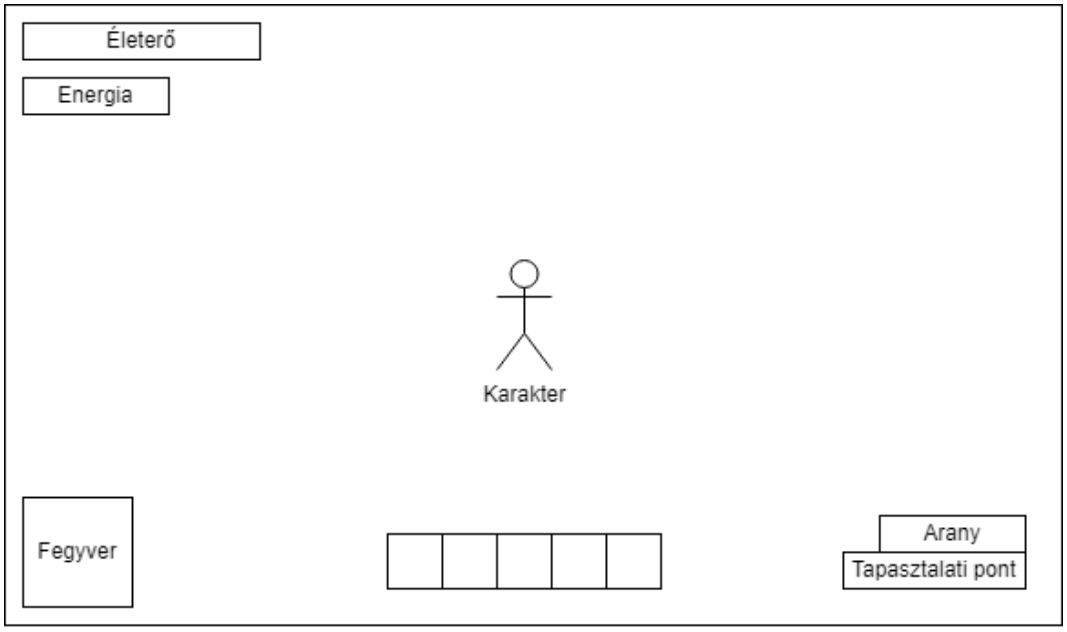
A játékokban fontos a jól megtervezett felhasználói felület amit, a játék közben lát a játékos, amely lehetővé teszi a játékosok számára, hogy a lehető legkönyebben és leggyorsabban elérjék a kívánt funkciókat. Ilyen felület a HUD (Head Up Display), amely segíti a játékosokat abban, hogy valós idejű információkkal rendelkezzenek a játék világáról és saját karakterük aktuális állapotáról, így gyorsabban és hatékonyabban reagálhatnak a különböző helyzetekre.

3.4.1. Felhasználói felület

A felhasználói felület megtervezése során nagyon kell figyelni a letisztultságra és a felhasználói élményre. Fontosnak tartottam, hogy a design egyszerű és könnyen átlátható legyen, hogy a felhasználó könnyedén megtalálja azokat a funkciókat és információkat, amelyekre szüksége van.

A Head Up Display (HUD) a játékos karakterének életpontjait, energia szintjét, a

játékos aranyát, a játékos szintjét, a játékos tapasztalati pontjait, a játékos fegyverét, a játékos által használható italokat fogja megjeleníteni. (Lásd 3.3 ábra) A HUD a játék során folyamatosan látható lesz, így a játékosoknak nem kell külön menübe navigálniuk, hogy megtalálják a folyamatosan szükséges és releváns információkat és cselekvési lehetőségeket.



3.3. ábra: Felhasználói felület terve

3.4.2. Életpontok és energia

A játékos karakterének életpontjai mutatják meg, hogy mennyi sebzést tudnak még elviselni, mielőtt meghalna a karakterük. Az energia szint pedig a karakter futásához és cselekvéseihez szükséges energiaforrás. A megfelelő kezelése és felhasználása kulcsfontosságú a túléléshez és a harc hatékonyságához.

3.4.3. Fegyverválasztás

A játékosnak lehetősége van választani számos közelharci fegyver közül, amelyeket a preferenciája és strategiaja alapján szabadon választhat meg. A fegyverek különböző sebzést okoznak, de ez függ a hatótávolságuktól is. Hiszen a rövidebb hatótávolságú fegyverekkel közelebb kell kerülni az ellenségekhez, hogy sebzést okozzon a játékos, ez nagyobb kockázattal jár ezért, azokkal több sebzést lehet okozni.

3.4.4. Harcolás szörnyekkel

A játék során a játékosnak számos szörnnyel kell megküzdenie. A harcok izgalmasak és változatosak, például a varázsló tud a pálcájából lövedékeket lőni a karakter irányába, amelyeket olykor nehéz kikerülni. A játékosnak ki kell használnia a karaktere képességeit és fegyvereit a sikeres küzdelem érdekében. A szörnyek legyőzése tapasztalati pontokat (XP) és esetleges zsákmányt is eredményez.

3.4.5. Küldetések felvétele

A lineáris történetvezetés lehetőséget kínál arra, hogy a játékosok fokozatosan merüljenek el a játék világában, miközben követik a fő cselekményt. A küldetések integrálása a lineáris narratívába lehetővé teszi, hogy a játékosok szorosan kövessék a történet fő vonalát, miközben változatos kihívásokkal találkoznak.

A küldetések kiváló lehetőséget nyújtanak a karakterfejlődésre és a történet gazdagítására. Az XP (tapasztalati pont) és jutalmak rendszere ösztönzi a játékosokat, hogy aktívan részt vegyenek a küldetésekben, és ezáltal fokozatosan erősödjenek a játékban. A lineáris elbeszélésmód segítségével a küldetések sorrendje és nehézségi szintje könnyen szabályozható, így biztosítható a folyamatos kihívás és az érdekesség fenntartása.

3.4.6. Bájitalok vásárlása és használata

A játékos aranyat szerezhet a játék során, amit bájitalokra is fordíthat. Azok különböző hatásokkal rendelkeznek, például sebzés gyógyítása vagy energiaszint növelése. A bájitalok stratégiai használata a túlélés és a harcok kulcsa lehet.

3.4.7. Statisztikák növelése tapasztalati pontokból

A játékos karaktere tapasztalati pontokat szerez a harcok és küldetések során. Az XP felhasználható a karakter statisztikáinak növelésére, például az erő, az ügyesség vagy a mágia terén. Ez lehetővé teszi a karakter fejlődését, specializálását és ezzel a játékmenet testreszabását.

Ezen cselekvési lehetőségek együttesen teremtenek egy gazdag és dinamikus játékélményt az Action RPG (akció-szerepjáték) játékban, ahol a játékos döntései és cselekedetei hatással vannak a karakter fejlődésére és a játék világára.

4. fejezet

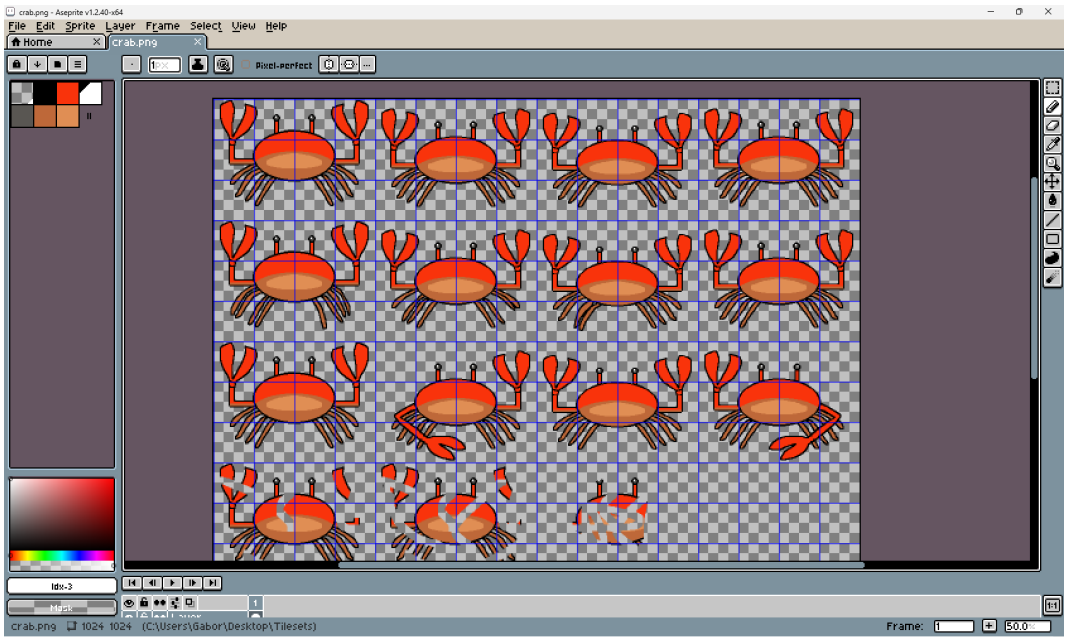
Megvalósítás

4.1. Grafikák elkészítése

Mielőtt nekiláttam a grafikai elemek létrehozásának, inspirációként szolgáltak az interneten elérhető rajzolási stílusok és csempekészletek. Az alapja és mintája a saját rajzaimnak a Ninja Adventure Tileset volt, amelyet Pixel-boy készített. [25]

4.1.1. Grafikus szerkesztő

Fontos kiemelni, hogy egy „pixelart” rajz stílusról beszélünk az esetemben, ezért kerestem olyan rajzprogramot, amely minden igényemet kielégíti és könnyen kezelhető tapasztalatlan grafikusok számára is. A kiválaszott program az Aseprite volt, ebben készítettem minden játékban megtalálható grafikát. (Lásd 4.1 ábra)



4.1. ábra: Aseprite - Pixelart rajzoló program [26]

4.1.2. Rajzok elkészítése

A rajzok elkészítése két fő részből állt, a pálya tervezéséből és a karakterek, valamint a környezet kialakításából. A pálya tervezéséhez a korábban említett Ninja Adventure Tileset szolgált mintaként, az összes pályaelemet saját kezűleg rajzoltam meg. A karakterek és környezet rajzainak elkészítésénél pedig különböző rajzokat használtam inspirációként, és ezek alapján készítettem el a saját rajzaimat.

Pálya

A pályaelemek létrehozásánál kiemelkedően fontos volt figyelnem arra, hogy a csempék (lásd 4.2) harmonikusan illeszkedjenek egymáshoz, ezáltal kellemes és összefüggő látványt teremtve.

4.2. ábra: Csempekészlet

Entitások

Karakterek és szörnyek (lásd 4.3) megtervezése annyiban tért el, hogy ott figyelni kellett az animációra is, mert élethű mozgást szerettem volna utánozni. Egy ilyen mozgás animáció 4-6 képből áll, amelyeket egymás után lejátszva érhető el a kívánt hatás. Illetve, mivel 4 különböző irányba tudnak haladni ezek az entitások, ezért 4 különböző animációt kellett elkészítenem, amelyeket a játék során a karakter irányának megfelelően tudtam lejátszani, ez látható a 4.1 ábrán.

4.3. ábra: Entitások

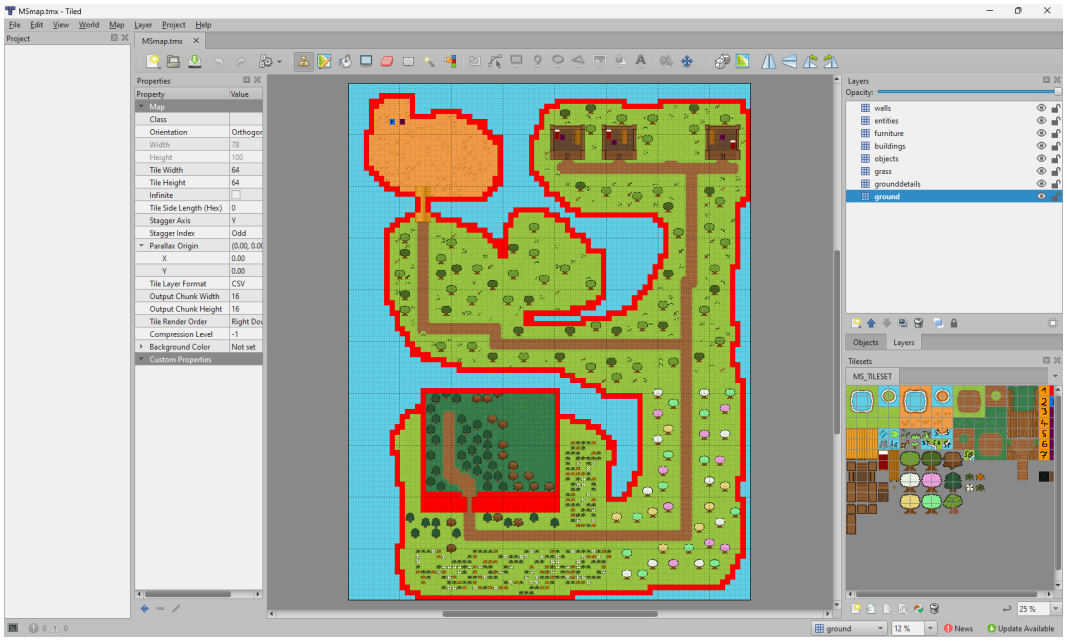
4.2. Pálya megtervezése

4.2.1. Tiled

A pálya megtervezéséhez a Tiled-ot [27] választottam, ami egy népszerű térképszerkesztő szoftver, amelyet játékfejlesztők használnak a 2D-s játékok pályáinak tervezésére és szerkesztésére. A Tiled egy nyílt forráskódú program, így ingyenesen elérhető és széles körben használt az indie játékfejlesztők körében.

Ez a szoftver lehetővé teszi a felhasználók számára, hogy könnyen létrehozzanak és testreszabjanak térképeket, amelyeket különböző játékkeretrendszerekbe integrálhatnak. A Tiled (Lásd 4.4 ábra) számos funkciót kínál, mint például csempekészletek használata, rétegek kezelése, objektumok lehelyezése rácshálóra, ezzel elkerülve a rétegen belüli átfetéseket. Valamint az exportálás különböző formátumokban, például CSV (Coma-Separated Values) vagy TMX (Tiled Map XML).

Az egyszerű és intuitív felhasználói felülete miatt a Tiled ideális eszköz a játékfejlesztők számára, akik az apró részletekre is figyelő térképeket szeretnének készíteni játékaikhoz. Az XML alapú TMX formátum kompatibilis a legtöbb játékmotorral, így a Tiled segítségével készült térképek könnyen beilleszthetők a fejlesztési folyamatba.



4.4. ábra: Tiled - Pálya szerkesztő [27]

4.2.2. Tervezés

Kiemelkedően fontos volt a pálya elemeit különböző rétegekre szétválasztani, ezáltal megkönnyítve a fejlesztést és az egyes részek kezelését. A háttér szolgálatában álló csempéket például két rétegre osztottam: az egyikre a talajszintet, a másikra pedig a talajon megtalálható növényzetet helyeztem el. Ezt a két réteget együttesen exportáltam PNG formátumban, amelyet a játékban háttérképként használtam. Fontos megjegyezni, hogy a két háttér réteget nem mentettem külön CSV fájlba. Ennek oka az, hogy ezek a rétegek olyan elemeket tartalmaznak, amelyek statikusak, és nincs velük semmiféle interakció a játék során.

Van még egy fontos réteg, amik akadályként szolgálnak a játékos számára (Lásd 4.4 ábra piros körvonal), ezt a réteget a játékos nem tudja átlépni, ezáltal a játékmenetet befolyásolják.

További rétegek lettek létrehozva az objektumok számára, minden típusú objektumot külön rétegen helyeztem el, így könnyen kezelhetőek, és nem kell a kódban szűrést alkalmazni a különböző típusú objektumok használatakor. Réteget hoztam létre a játék során kiüthető objektumoknak, ahova jelenleg a fűcsomók tartoznak. Nem kiüthető objektumoknak, mint például házak vagy fák. Berendezési tárgyaknak, mivel azok szintén objektumok, mert szerettem volna, hogy a karakterek át tudnak sétálni azokon. Illetve nem utolsó sorban az entitásoknak, hogy milyen pozícióra rajzolja ki őket a játék. A rétegeket a 4.4 ábrán a jobb oldalon látható ablkaban lehet kezelni.

4.3. Felhasználói felület és beállítások kezelése

4.3.1. Beállítások

A funkció fő célja a beállítások kezelése egy játékban vagy alkalmazásban. Az osztály tartalmazza az alapértelmezett beállításokat, mint például a játékablak méretét, a hangerejét és egyebeket. Az osztály segítségével ezeket a beállításokat be lehet tölteni vagy menteni egy JSON fájlba.

A statikus osztályváltozók az alapértelmezett értékeket tárolják, és a program különböző részeiben felhasználhatók, anélkül, hogy az osztalyt peldanyositani kellene, így könnyen hozzáférhetőek.

További adatokat is tartalmaz, például hangokat, grafikákat, karaktereket, ellenségeket, lövedékeket és fegyvereket. Mindezek az adatok részletezik a játékbeli objektumok tulajdonságait. Azért ezt a megoldást választottam, mert így könnyen kezelhetőek és könnyen módosíthatóak a játékbeli objektumok tulajdonságai. Egy helyen kezelve egyszerűbb a karbantartás is, ha egy szörny túl erős lenne vagy egy fegyver túl gyenge, akkor csak egy helyen kell módosítani az adatokat, és az összes példányban azonnal érvényesülnek a változtatások.

Az osztály emellett definiálja a felhasználói felület színeit és stílusait, amelyeket a játék megjelenítéséhez használok. Ez azért fontos, hogy egységes megjelenést biztosítson a felhasználói felületben, és ahogy az előbb is említettem, egy színkód megváltoztatásával mindenhol azonnal érvényesül a változtatás.

4.3.2. Felhasználói fiók

Ez a rendszer kezeli a regisztrációhoz és a bejelentkezéshez kapcsolódó folyamatokat. Ilyen folyamat többek között a felhasználói felület megjelenítése, de a tartalomellenőrzés és hibakezelés is. Tartalomellenőrzés alatt a felhasználónév és jelszó validálás értendő, hogy megfelelő formátumban vannak-e megadva. Hibakezelés alatt pedig a felhasználó értesítése, ha valamilyen hiba történik a regisztráció vagy bejelentkezés során. Regisztráció esetén, ha sikeres validálás történik, akkor md5 hash algoritmus segítségével titkosítva tárolja a jelszót az adatbázisban. Bejelentkezéskor ezt a titkosított jelszót hasonlítja össze a felhasználó által megadott jelszó titkosított formájával, ha megegyezik, akkor engedi a bejelentkezést.

4.3.3. Főmenü

A játéknak elengedhetetlen része a főmenü, ami a bejelentkezés után válik elérhetővé a felhasználók számára. Biztosítok a játékosoknak offline, azaz internet nélküli játszási lehetőséget, ilyenkor regisztráció és bejelentkezés nélkül használni lehet a játékot. A főmenüben legfelül a játék címe „Marooned Sailor" jelenik meg, alatta három menüpont található meg, az új játék létrehozása, meglévő mentés betöltése, illetve a beállítások. (Lásd 4.5 ábra) Kijelentkezési lehetőséget nem valósítottam meg, ahogy kilépés gombot sem helyeztem el, ezzel szeretném ösztönözni azokat akik kipróbálják, hogy ne zárják be első látásra a játékot. Bezárni az asztali alkalmazást a pillanat állj menüben lehet majd a játék során az exit menüpontra kattintva.

4.5. ábra: Főmenü

4.3.4. Új játék létrehozása

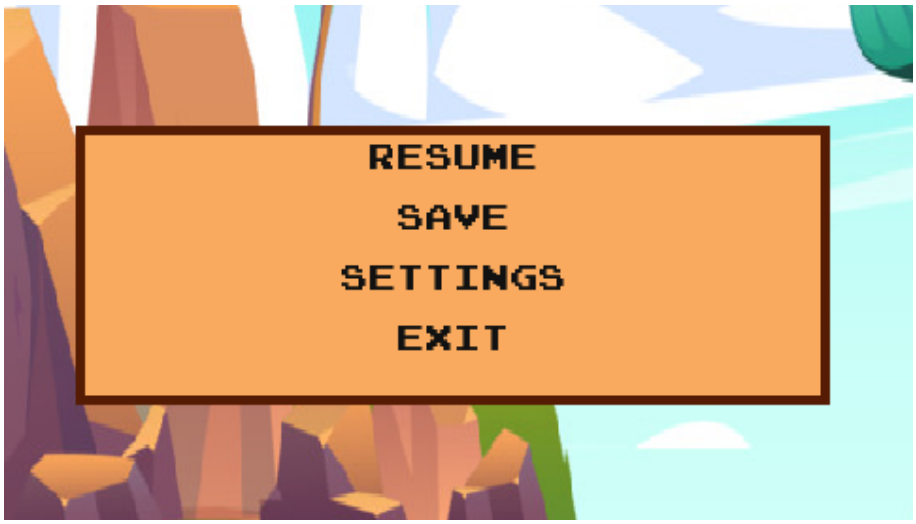
Az új játék menüpont kiválasztása alatt (lásd 4.6 ábra) a játékosnak meg kell adni a karakterének a nevét, illetve választhat különböző karakter kinézetek közül. Ha be van jelentkezve a játékos, akkor van lehetősége nehézségi szintet is választani. A nehézségi szint annyiban különbözik, hogy a „normal" módban, ha elfogynak a játékos életpontjai, akkor a játék folytatódik tovább, újraéled egy bizonyos helyen. Viszont, ha a „challenge" (kihívás) módot választja, akkor három funkcióval bővül a játékmenet. Az első ilyen funkció, hogy ha elfogy a játékos életereje, akkor véglegesen meghal a karakter, és nem lesz lehetősége tovább folytatni azzal a bizonyos karakterrel a játékot. A második plusz funkció, hogy az összes küldetés teljesítése esetén a játék újrakezdődik, amely azt eredményezi, hogy a szörnyek erősödnek, és nagyobb kihívás lesz újra végigvinni az összes küldetést. Illetve tartalmaz egy ranglista menüpontot, amely jelzi, hogy a legügyesebb játékosok hányszor tudták a „challange" mód használatával végigvinni az összes küldetést.

4.6. ábra: Új játék létrehozása

4.3.5. Játékmenet közbeni menü

Az Esc gomb megnyomásával elérünk egy pillanat állj funkciót, amely egyben egy menüt is megjelenít (Lásd 4.7 ábra), ahol a játékosnak lehetősége van a játékállapot mentésére, illetve folytatni a játékot vagy kilépni az asztalra.

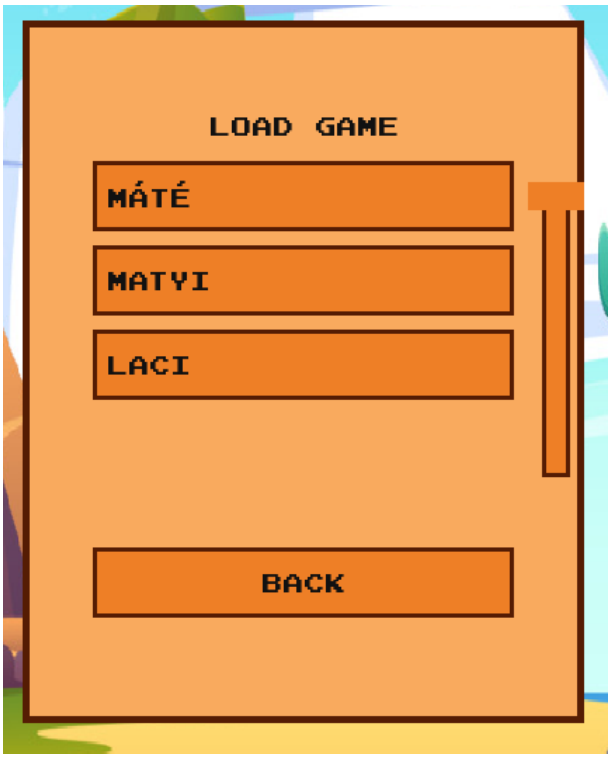
A pillanat állj funkciót úgy valósítottam meg, hogy az összes játékbeli felületcsoport, (később bővebben lesz szó ezekről a 4.4.4 szakaszban) kirajzolására szolgáló függvényt állítom le egy boolean típusú változóval, ezáltal minden játékon belüli elem megáll a legutolsó ismert állapotában, majd a menü bezárása után, amelyet a resume (folytatás) gomb megnyomásával tehet meg a játékos, onnan folytatódnak az események, ahol abbamaradtak.



4.7. ábra: Játékmenet közbeni menü

4.3.6. Mentések betöltése

A játékosnak lehetősége van a korábban elmentett játékállás betöltésére is, amennyiben van ilyen. A betöltés után a játék onnan folytatódik, ahol abbahagyta. Egy lista elrendezésben látja a felhasználó a korábbi mentéseit. (Lásd 4.8 ábra) Előre rendezi az offline mentéseket a listában, és csak utánuk tölti be az online mentéseket. Egy kattintás után már be is tölti az adott játékállást. Külön nem jelzi, hogy az adott mentés melyik kategóriába tartozik, egyszerűen nem jelennek meg a listában az online mentések, ha nincs bejelentkezve a felhasználó.



4.8. ábra: Mentés betöltése

4.4. Játékmenet kezelése

4.4.1. Main metódus

Ez az osztály felelős a játék főciklusának végrehajtásáért, ami az egész játék működésének alapját képezi. Amikor a játék elindul, az alkalmazás példányosítja ezt az osztályt.

A 4.1 programkód részletesen bemutatja, hogyan kell megvalósítani ezt a főciklust, amely a játék működéséért felelős. Ebben a kódban kezeljük a játék fő eseményeit, és gondoskodunk arról, hogy minden szükséges műveletet végrehajtsunk a játék folyamán.

Amikor elindul a játék, a bejelentkezési képernyőn találjuk magunkat. A menuenums enum típusban tárolt értékek segítségével dönti el a program, hogy mely állapotot kell megjeleníteni. Itt gondolok, a bejelentkezési képernyőre, főmenüre és annak almenüire, illetve a játékra magára.

A főciklusban történik:

- a képfrissítés mértékének beállítása a self.clock.tick(Settings.FPS) sorral.

- az ablak háttérszínének beállítására a self.screen.fill(Settings.WATER_COLOR) sorral.
- a játékmenet futtatására a self.game_handler.run() sorral.
- a képernyő frissítésére is a pygame.display.update() sorral.

Tehát létfontosságú szerepet játszik abban, hogy a játék működése zökkenőmentes legyen, és a 4.1 programkódban látható példa segítségével könnyen megérthető, hogyan történik mindez.

Programkód 4.1. Játék főciklusa

```
def run(self):
    while True:
        [...]
        elif self.state == menuenums.GAME:
            if self.game_handler is None:
                Sounds.play_loop('main')
                self.game_handler = GameHandler(
                    self.user, self.save_parameters)
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()
            self.screen.fill(Settings.WATER_COLOR)
            self.game_handler.run()
            pygame.display.update()
            self.clock.tick(Settings.FPS)
```

4.4.2. Játékon belüli felületek

A GameHandler osztály nagyon fontos szerepet tölt be a játék futása során, több alapvető funkciót kezel. Ez az osztály kezeli a mentéseket, erről a későbbiekben a 4.4.3 szakaszban tárgyalunk majd. Kezeli a megjelenített felületeket, értem ezalatt a játékmeneten belüli menüt, ez valósítja meg a pillanat állj funkciót is, mivel megállítja a LevelHandler (erről a 4.4.4 szakaszban lesz szó) álltal frissített felületeket. Itt kezelt felületek a különböző felugró ablakok, például a ranglista funkció, amely a játékmenet megállítása nélkül is látható. Az utolsó általa kezelt felület a karakter fejlesztési panel is, amely a játékos karakterének fejlesztésére szolgál.

4.4.3. Játékállapot mentése

A játékállapot elmentését Save gomb megnyomásával lehet elérni. A mentés a játékállapot minden fontosabb elemét eltárolja egy JSON típusú fájlban, ha offline módban játszik a játékos, vagy FastAPI-on [29] keresztül kommunikálva egy MySQL [30] adatbázisban, ha online módot választott a felhasználó. Minden karakterhez egy mentési lehetőség tartozik, tehát nem lehet egy karakterhez több játékállapotbeli mentést készíteni. Mindig az utolsó mentés kerül felülírásra, ha a játékos új mentést készít. Ebből következik, hogy a játékosnak mentéskor nincs lehetősége rálátni a korábbi mentéseire, csak a betöltési menüben, ahol a játékos kiválaszthatja a korábbi mentéseit.

4.4.4. Pálya kezelése

LevelHandler a másik nagyon lényeges osztály, ez a GameHandler-ben kerül példányosításra. Feladata a világok, felületek, kamera és entitások létrehozása és kezelése. Ellentétben a GameHandler osztállyal (Lásd 4.4.2 szakasz) amely feladata a játékon belül elérhető funkciók kezelése, értem ezalatt a menük megjelenítését, mentést vagy a pálya futtatását. A különböző felületeket Sprite Group-okba (felületcsoport) szervezve kezeli, így könnyen tudja őket elérni és frissíteni. Ezeket a csoportokat a Tiled-ban létrehozott rétegek neve alapján határozza meg a prgogram, milyen felület milyen jellemzőkkel rendelkezzen. Legfontosabb ezek a felületcsoportok közül talán a Visible felületcsoport, amely a játékos által látható felületeket tartalmazza. A kamera ezeket rendereli ki úgy, hogy ezeket a csempéket y tengely szerint rendezi és úgy jeleníti meg, hiszen, ha az alapértelmezett x tengelyen haladna végig a kirajzolási funkció, akkor nem lehetne elérni hamis 3D hatást. Jelenleg, ha a karakter egy fa „felett" tartózkodik, azt félig rá rajzolja ki, ezáltal elérve, mintha mögötte bújkálna a karakter, de ez minden elérhető felületre igaz. Maradva az előző példánál, ez x tengely szerinti rendezés esetén úgy nézne ki, hogy a karakter a fa baloldalánál állva a fa mögött helyezkedne el, azaz a karaktert kitakarná, ameddig a jobb oldalán a karakter lenne az előtérben. Megtalálható, még obstacle (akadály), attack (támadó), és attackable (támadható) felületcsoportok is, amelyek a játékmenet során fontos szerepet játszanak, különböző mechanikák végrehajtásánál. Obstacle csoportot a 4.4 ábrán piros körvonalú rétegen szereplő csempék kaptak, amelyek szerepe, hogy az ütközésvizsgálat (később a 4.9 szakaszban írok erről bővebben) az ebbe a csoportba tartozó csempéket veszi figyelembe. Az attack csoportba a játékos által irányított karakter támadó felületei kerültek, azaz a fegyverek, hiszen ezeket nagyon rövid ideig kell megjeleníteni, ezután törölni azokat. Az attackable csoport szorosan kapcsolódik az utóbb említett csoporthoz. Ide tartozik az összes olyan felület, amelyekkel a fegyverek interakcióba kerülhetnek. Ebbe a csoportba tartoznak például a szörnyek, de a fűcsomók is, amelyeket a játékos ki tud ütni a fegyverével.

4.4.5. Világ és barlangok

Amikor a játékos létrehoz egy új karaktert, vagy betölt egyet, akkor egy világban (world) fogja találni magát. Ez a világ a fő pályarész, itt helyezkedik el minden kulcsontosságú helyszín és karakter. Illetve a játék során el lehet látogatni egy barlangba (dungeon), ami egy nagyságrendileg kisebb terület, amelyből több is létezhet egy világban. A world és a dungeon osztály közös szülő osztályból származnak le, ez azért fontos, hogy a különböző világokat könnyen tudjam kezelni, de közös tulajdonságokat adni nekik. Ebben az osztályban a minden világra vonatkozó lehetőségeket kezelem, ilyen például az effekteket lejátszó AnimationPlayer osztályom származtatása, mert azt elég egyszer definiálni és meghívni ott, ahol szükséges, ahelyett, hogy minden világnak lenne egy külön példánya. Továbbá ilyen opció a térkép megnyitása, és az azzal kapcsolatos összes funkció, például, hogy hol található a következő küldetést adó NPC (nem játékos karakter). A játék története jelenleg teljesen lineáris, tehát egy adott sorrendben történhet csak a küldetések teljesítése. Ez az osztály kezeli a szörnyek legyőzése után elejtett, hátrahagyott tárgyakat, ilyen tárgyak a tapasztalati pont és az aranypénz. Ezeket a tárgyakat a játékos össze tudja gyűjteni, és a későbbiekben fel tudja használni.

Pálya generáláshoz szükséges adatok

Említettem a korábbi fejezetben, a pálya megtervezését a **Tiled**-al végeztem. (Lásd 4.2.1 szakasz) Első megközelítésemben CSV (vesszővel elválaszott értékek) fájl formátumba mentettem ki a tervezőben létrehozott rétegeket. Egy ilyen dokumentum úgy néz ki, hogy ahol nem helyeztem le a tervezőben objektumot, ott '-1' érték szerepel, ellentétben ahol van érték, ott a Tiled-ben betöltött és feldarabolt csempekészlet azonosító számai kerültek a dokumentumba. Ezt úgy kell elképzelni, mint egy nagy mátrixot. Az utolsó pillanatokig ezt a megközelítést alkalmaztam, mert ha már kész volt és működött, miért változtattam volna rajta. Végül mégis visszatértem erre a kérdéskörre és készítettem az alább olvasható kis egyszerű átalakító szkriptet. (Lásd 4.2 programkód) Ennek funkciója a sok felesleges '-1' értéket kiszűrni, hogy a program futása közben ne kelljen a betöltésnél rengeteg felesleges összehasonlítást végezni. Ezt úgy oldottam meg, hogy a készítettem Python dictionary-t '-1' értékeket elhanyagolva, a csempe azonosító számából, illetve annak a .CSV mátrixban elhelyezkedő koordinátáiról. (Lásd 4.3 programkód)

Programkód 4.2. CSV formátum dictionary formátumra konvertálása

```
import csv
import json

csv_file = "new_map\MSmap._dungeonentrance.csv"
map_data = []

with open(csv_file, newline='') as csvfile:
    csv_reader = csv.reader(csvfile)

    for i, row in enumerate(csv_reader):
        for j, value in enumerate(row):

            value = int(value)
            if value != -1:
                map_data.append({"i": i, "j": j, "value": value})

python_file = "world_dungeonentrance"

with open(python_file, 'w') as pyfile:
    pyfile.write("map_data = ")
    json.dump(map_data, pyfile, indent=4)

print(f"Dictionary saved to {python_file}")
```

Készítettem egy kis függvényt, a pálya létrehozásának idejének lemérésére. Ez azért volt fontos, hogy meg tudjam határozni, hogy a szótár használata jobb megoldás-e, mint a CSV fájl. A két megoldás esetében a következő eredményeket kaptam:

- Python dictionary esetében: `Futási idő: 108.64 ms`
- Comma-Separated Values (CSV) esetében: `Futási idő: 879.92 ms`

Tehát elmondható, hogy ez a változtatás jelentősen felgyorsította a pálya létrehozását, ezáltal a játék betöltési ideje is csökkent.

Programkód 4.3. Minta az átalakított struktúrára

```
{
    "i": 3,
    "j": 41,
    "value": 125
},
```

Pálya generálás

A korábban (lásd 4.2.2 szakasz) megtervezett és exportált majd feldolgozott (Lásd 4.4.5 szakasz) pályaadatok megjelenítését több lépésben valósítottam meg. Először is a képeket be kell tudnom olvasni, ezt az os [28] könyvtár walk függvényének segítségével oldottam meg. Miután sikerült bejárni a mappát, előállította az egyes képekhez vezető útvonalat, ezután a pygame.image.load() függvényével betölti a képeket. A képeket entitás esetében képkockánként, míg a pálya esetében mozaik-kockák formájában tároltam. Ezt követően a képeket egy listába helyeztem, amelyet később fel tudok használni a kirajzoláshoz. Másik ilyen fontos lépés a pálya rétegek beolvasása, amelyet a korábban említett dictionary segítségével valósítottam meg. Ezután egy ciklusban végig iterálva a dictionary-n, a képek listájából kiválasztotta a megfelelő képet, és a megfelelő koordinátákat, amelyek segítségével példányosítom a Tile osztályt, később a program ennek segítségével végzi el a kirajzolást. Mivel a szótárban szerepel az entitásokra vonatkozó réteg is, ezért azokat is az előbb említett iterációban kezeltem, és példányosítottam a hozzájuk tartozó osztályokat, azonosító alapján szűrve.

4.4.6. Entitások

Ez az osztály egy szülő osztály, amit az Enemy, NPC és Player osztályok használnak. A szülő osztály azt jelenti, hogy ebből származnak le más osztályok és felhasználják minden tulajdonságát, illetve metódusait is.

Három fontos metódussal rendelkezik, az egyik a mozgás kezelése. A mozgást a move függvény segítségével hajtja végre, amelynek a lényege, hogy az entitás rendelkezik egy irány vektorral,és annak normalizálásával, illetve a paraméterként megkapott sebesség értékkel meghatározza a következő pozícióját az entitásnak. A normalizás azért fontos, hogy átlós mozgás esetén ne mozogjon az adott karakter sokkal gyorsabban. A vektor egységesítéséhez, normalizálásához a vektor minden egyes komponensét el kell osztani annak hosszával, ezzel elérve, hogy a vektor iránya ne változzon, csak a hossza legyen 1. A pozíció módosítása az egyed hitbox-ára vonatkozik, azaz arra a dobozra, amely az entitást körülveszi. Azért a hitbox-ot kell mozgatni, mert a szörnyek és karakterek képei a hitboxra vannak igazítva. Ez azért fontos, mert így megelőzhető az olyan hibák előfordulása, amit a hitbox és a kép közötti eltérés okozhatna. Ilyen szinkrinizációs hiba, például lövöldözős játékok esetében szokott látványosan előfordulni, hogy egy bizonyos testrész hitbox-a nem ott helyezkedik el, ahol a grafika, és bár a játékosok látják, hogy eltalálják az ellenfelet, mégsem történik sebzés.

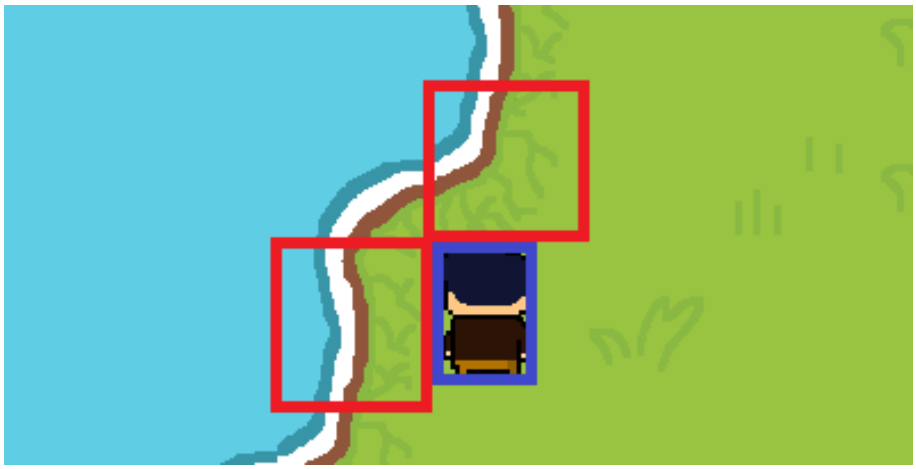
Sebzőséd után az entitások sebezhetetlenek egy kis ideig, és ennek a jelzésére szolgál a második metódus. Ez egy szinusz függvény alapján ad vissza 0-255 értéket, amellyel

az entitások áttetszőségét állítom be.

A harmadik metódus az ütközés érzékelésre, kezelésére szolgál. Az ütközés érzékelés lényegében úgy működik, hogy nézzük az entitás mozgási irányát, hogy függőlegesen, vagy vízszintesen közlekedik, és az adott iránytól függően vizsgáljuk, hogy az entitás fizikai kiterjedése (későbbiekben hitbox), esetemben egy grafika nagyságú téglalap ütközik-e valamelyik korábban említett obstacle, akadály csoport csempével. Ha igen, akkor az entitás irány szerinti hitbox határát az akadály ellenkező irányű hitbox határához igazítjuk, így megakadályozva az áthaladást, de nem letiltva a többi irányba való mozgást. (Lásd 4.9 ábra és 4.4 programkód)

Programkód 4.4. Ütközés kezelése

```
def collision(self, direction):
    #horizontalis mozgas
    if direction == 'horizontal':
        for sprite in self.obstacle_sprites:
            if sprite.hitbox.colliderect(self.hitbox):
                if self.direction.x > 0:
                # jobbra mozgas
                    self.hitbox.right = sprite.hitbox.left
                if self.direction.x < 0:
                # balra mozgas
                    self.hitbox.left = sprite.hitbox.right
    #vertikalis mozgas
    if direction == 'vertical':
        for sprite in self.obstacle_sprites:
            if sprite.hitbox.colliderect(self.hitbox):
                    if self.direction.y > 0:
                        # lefele mozgas
                        self.hitbox.bottom = sprite.hitbox.top
                        if self.direction.y < 0:
                        # felfele mozgas
                        self.hitbox.top = sprite.hitbox.bottom
```



4.9. ábra: Ütközés kezelése

4.4.7. Játékos karaktere

Ez az osztály a játékos karakterét valósítja meg, amely a játék során irányítható. A játékos karaktere egy entitás, így örökli az entitás osztály összes tulajdonságát és metódusát.

Az osztály inicializálása során számos fontos adatot tárol, például a játékos nevét, karakterének azonosítóját és kezdeti pozícióját. Ezek az adatok meghatározzák a karakter kezdeti állapotát a játékban.

A karakter mozgását és irányítását az osztály input metódusa kezeli. Ennek segítségével az osztály figyeli, hogy milyen billentyűk vannak lenyomva, és ennek megfelelően mozgatja a karaktert a játékban. A játékos karakter képes balra, jobbra, felfelé és lefelé mozogni a megfelelő billentyűk lenyomásával.

A karakter sebessége és energiaszintje dinamikusan változik a játék során. Például a karakter gyorsabban mozoghat, ha lenyomja a „SHIFT" billentyűt, és az energia szintje csökken mindeközben. Az energia idővel regenerálódik, ami lehetővé teszi, hogy tovább használhassa a gyors futás funkciót.

A játékos karakter képes támadni is, és az attack metódus meghívásával hozza létre a támadást. A támadásokat az osztály figyeli, és számítja ki a támadások között eltelt időt, hogy ne lehessen túl gyorsan ütni a fegyverekkel.

A karakter fegyvere (bővebben a 4.4.8 szakaszban tárgyalunk majd) is dinamikusan változik, és a játékos lehetőséget kap a fegyver cseréjére a játék során. Fegyvert a tabulátor gomb segítségével harc közben is választhat.

A karakternek vannak életpontjai, amelyek meghatározzák a túlélését a játékban. Amennyiben az életpontja nullára csökken, a karakter meghal, de van lehetőség a visszatérésre vagy újrakezdésre a játékban, attól függően, hogy milyen a játék nehézségi szintje.

A karakter számos egyéb tulajdonsággal és képességgel rendelkezik, például tapasztalati pontokkal, megszerzett arany mennyiségével és teljesített küldetésekkel. Ezek a tényezők befolyásolják a játékmenetet és a karakter fejlődését.

Az osztály továbbá lehetővé teszi a karakternek, hogy tárgyakat használjon és rendelkezzen egy táskával. A játékos képes tárgyakat váltani és használni a játék során, ami további taktikai elemet ad a játékhoz.

Összességében ez az osztály kulcsfontosságú a játékos karakter kezelésében és irányításában, lehetővé téve a játékosnak, hogy részt vegyen a játék világában és kihívásokkal nézzen szembe a karaktere fejlesztése során.

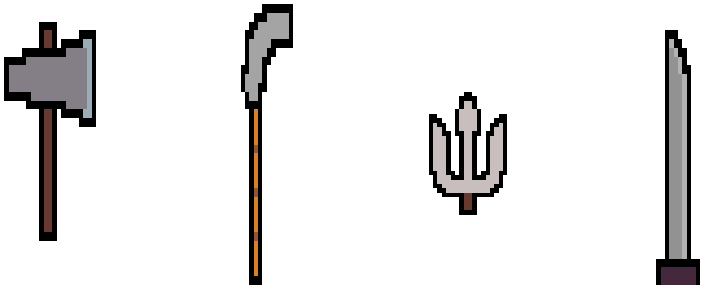
4.4.8. Fegyverek

A különböző fegyverek (Lásd 4.10 ábra) használata szintén egy elengedhetetlen része a játéknak. A fegyvereket a játékos karaktere használja a szörnyek elleni harc során. Közlehraci fegyvereket valósítottam meg, a kard, a lándzsa, a kétkezes kard és a kétkezes lándzsa. Ezek a fegyverek különböző támadási távolsággal és sebzéssel rendelkeznek, előkészítettem a támadási sebesség megvalósítását is, de az végül nem került be a játékba, mert szerintem rontotta volna a játékélményt.

Ezen harcolásra szolgáló eszközök kirajzolása a támadás gomb lenyomása után történik a Weapon osztály példányosításával. Ez az osztály a játékos karaktere által kezében tartott fegyvert rajzolja ki a képernyőre. Mivel fontosnak tartottam, hogy mozgás közben is lehessen támadni, illetve legyen egy csapás/suhintás animáció, ezért egy frissítés funkciót vezettem be, amely mindig a játékos karakterének legutóbbi helyzetét,

és irányát vizsgálja, majd ezekre az adatokra alapozva frissíti a már létrehozott képet. A csapás animációt mind a négy irányban külön-külön kellett megtervezni, mert a fegyverek nem egyformán néznek ki minden irányból. Szinusz függvény segítségével valósítottam meg a megjelenített fegyver forgatását egy amplitúdó értékkel, amely a fegyver típusától függően változhat.

Ahogyan a többi látható objektum, ezek a fegyverek is a visible felületcsoport része, viszont egyben az attack felületcsoport része is, amely segít megkülönböztetni a többi felülettől a fegyvereket, ezáltal támogatva a támadási logika megvalósítását. A támadási logika egy metódus, amely ezen az attack felületcsoporton végig iterál, és vizsgálja az attackable felületcsoporttal való ütközést, amelyeket olyan entitások kaphatnak, amelyekkel a játékos meg tud küzdeni a játék során. Ez lehet egy szörny, vagy akár egy kiüthető fűcsomó.



4.10. ábra: A játékban megtalálható fegyverek

4.4.9. Ellenségek, szörnyek

Az ellenségek a küldetések után talán a legfontosabb mechanika a játékban, mert a velük való harc kiemelt szerepet kap a történet során. Az Enemy osztálynak is az Entity a szülő osztálya, ez biztosítja, hogy mozoghassanak a szörnyek. Szerettem volna, hogy az ellenségeket különböző tulajdonságokkal tudjam felruházni, de mégis egységesen tudjam kezelni. Például a sebesség, életerő, támadási sebesség, támadások hangjai. Ezeket a tulajdonságokat a settings statikus osztály monster_data dictionary-jében tárolom és olvasom be. Majd a szörnyek típusa alapján ezeket a tulajdonságokat beállítom az adott szörny példányosításakor.

Három különböző státusszal rendelkezhet egy ellenség. Ezek a státuszok határozzák meg a szörnyek viselkedését és a megjelenített grafikát is a játék során. A játékos karaktere, ha nincs az érzékelési sugarán belül, akkor nyugalmi státuszt kap. Mozgás státuszt kap, ha a játékos karaktere a szörny érzékelési sugarán belül van, de még nem támadási sugáron belül. Illetve támadási státuszt kap, ha támadási sugáron belül tartózkodik a játékos karaktere, ilyenkor lejátssza a hozzá tartozó támadási animációt, és sebzést tud okozni a játékos karakterének. Fontos, hogy csak bizonyos idő elteltével támadhat újra, elkerülve a hirtelen túl sok sebzés okozását.

A szörnyek mozgását úgy valósítottam meg, hogy meghatározom a játékos távolságát, és ha a távolság nagyobb, mint 0, azaz nem egy pozíción állnak, akkor meghatározom az irányt a két entitás pozícióinak különbségének normalizálásával. A távolság meghatározása azért fontos, hogy megállíptsam mikor ér a játékos karaktere a szörnyhöz támadási távolságon belülre. Az irány értéket pedig a entitások mozgatásáért felelős move metódusnak adom át paraméterként. (Lásd 4.4.6 szakasz)

4.4.10. Zsákmányolás

A zsákmányolás funkciót azért tartottam fontosnak, mert plusz motivációt nyújt a játékosnak legyőzni a szörnyeket, ha szerezhet tőlük különböző erőforrásokat. Egyenlőre a szörnyek kettő különböző erőforrás hátrahagyására képesek. Tapasztalati pont bogyókat hagynak hátra, amelyek előre meghatározott mennyiségű tapasztalati pontot tartalmaznak, mindegyik szörny ilyet hagy hátra a legyőzése után. Ezenkívül aranyat is hagyhatnak hátra, két különböző fajtát, bár erre már nem minden szörny képes, és ez ritkábban fordul elő. Az egyik fajta kis mennyiségű aranyat tartalmaz, míg a másik nagyobb mennyiségű aranyat ad.

A drop_loot függvényt a szörnyek példányosításkor paraméterként megkapják, és ha elfogy az életpontjuk, akkor hívják meg azt. Ez a metódus meghatároz a szörny legutóbbi pozíciójához viszonyítva egy 100x100-as négyzeten belüli pozíciót, ahol a zsákmányolható erőforrások fognak kirajzolásra kerülni. Ezt követően a szörny típusához tartozó loots listát kiolvassa a Settings osztály monster_data dictionary-jéből, hogy az adott szörny milyen zsákmányokat milyen eséllyel hagy hátra. Majd az előre meghatározott esély alapján véletlenszám-generálással eldönti, hogy milyen zsákmányokat hoz létre. A létrehozás úgy történik, hogy hozzá fűzi a Loot osztály egy példányát, amely tartalmazza tárgy nevét, mértékét és helyét, az adott pálya (World, Dungeon) példánynak a loots osztályváltozójához, ami egy alapértelmezetten üres lista.

A kirajzolása a felvehető tárgyaknak a visible felületcsoport frissítésével történik, mert a Loots osztály, ahogy a többi megjelenítendő objektum is, a pygame.sprite.Sprite osztályból származik le. (Lásd 4.11 ábra)

Tárgyak felvétele a tárgy pozíciója és a játékos karakterének távolságát vizsgálva történik, ha elég közel megy a játékos a tárgyhoz, akkor kitörli a loots listából a felvett tárgyat, majd az értékeit hozzáadja a játékos karakterének a tulajdonságaiban tárolt értékekhez. A hátrahagyott erőforrások addig a földön maradnak, ameddig a játékos fel nem veszi azokat. Ez a jelenlegi játékméretnél még nem okoz teljesítménybeli romlást.

4.11. ábra: Zsákmányolás funkció

4.4.11. Nem a játékos álltal vezértelt karakterek

Nem játékos által vezértelt karakterek, azaz NPC-k egy kalandjáték során elengedhetetlen részei a történetnek és a játékmenetnek. A szörnyek sem a játékos által vezérelt karakterek, de a játékokban az NPC megjelölést általában az egyéb, békés szerepet betöltő karakterekre használjuk. Ahogy az szörnyek és a játékos is az Entity osztályból származik le, úgy az NPC is, mert adott esetben szükség lehet mozgatásra és ütközés érzékelésre. Azonban a legtöbb esetben a játékos karaktere nem tudja megölni ezeket az entitásokat, mert a történet során szükség van rájuk. Ezért a játékos karaktere nem tud ütközni velük, és a szörnyekkel ellentétben nem tudnak támadni.

4.4.12. Küldetést adó karakterek

A játékom egyik legfontosabb mechanikája a küldetés rendszer, mert ezen alapszik az előrehaladás, illetve a challange játékmódnál az újraindítás is a küldetések teljesítését érzékelve valósul meg.

A Questgiver NPC (küldetést adó karakter) típust a World osztályban példányosítom, (Lásd 4.4.5 szakasz), és a példány osztályváltozóinak inicializálásakor olvassa be a saját id-jéhez tartozó küldetéseket a Settings osztály npc_data dictionary-jéből. Majd ezt követően a World osztály létrehoz a Player osztályból is egy példányt, amely egészen a LevelHandler osztálytól (amit már korábban említettem 4.4.4 szakaszban) kap paraméterként egy questgivers_quest_setup függvényt, amely egy trigger (aktiváló esemény) után vizsgálja, hogy a játékos álltal már teljesített küldetés szerepel még Questgivernél. Igaz visszatérési érték esetén kitörli azt, ezáltal elkerülhető a küldetés ismétlődése.

4.12. ábra: Dialógus

Az questgiver típusú NPC-k képesek egy előre megírt dialógus megjelenítésére (Lásd 4.12 ábra), amelyben a játékos választhat két lehetőség közül, amelyek a küldetés elfogadása vagy elutasítása. A választás megerősítése történhet a kurzornak a gomb felé mozgatásával, ezután egy kattintással, vagy a billentyűzet segítségével egyaránt. A dialógus ablak megnyitásához szükséges a játékos karakteréhez viszonyított távolság érzékelése, mert egy beszélgetés során közel kell állni a kommunikációban résztvevő többi szereplőhöz, ezt hasonló módon vizsgálom, mint ahogy a szörnyek esetében az

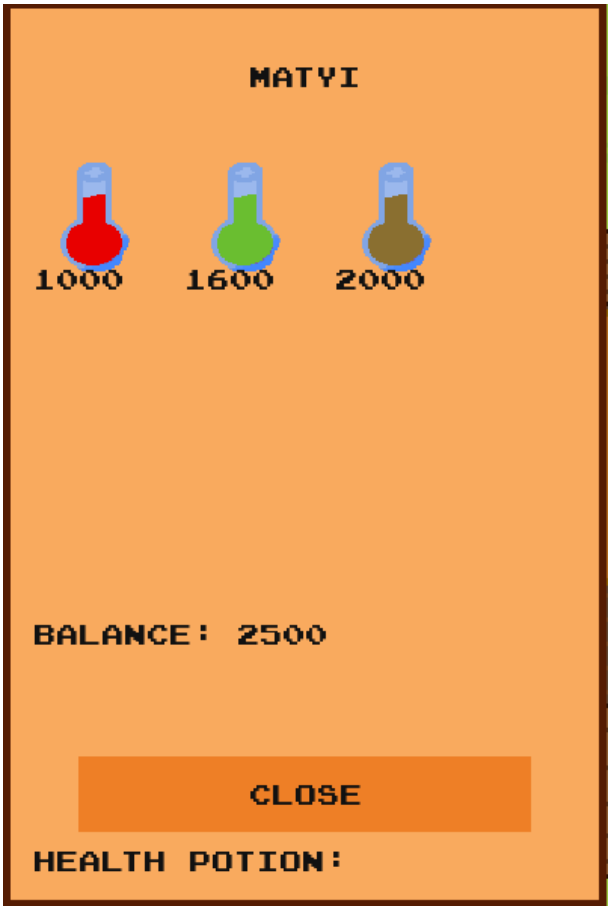
érzékelési távolságot vizsgáltam. Illetve dialógus megjelenítéséhez még szükséges, hogy az adott NPC rendelkezzen a soron következő küldetés azonosítójával is. Küldetést nem lehet kihagyni, mindegyiket sorban kell teljesíteni, ahogy a történet halad előre. Elutasítás esetén a küldetés nem törlődik, csak a dialógus ablak bezáródik, a játékban a linearitása miatt nem lehet tovább haladni.

4.4.13. Kereskedő karakter

Egy kalandjátékban az előrehaladás mellett fontos, hogy a nehezen megszerzett tárgyakkal, esetünkben az arannyal lehessen mit kezdeni, ezért egy Merchant NPC (kereskedő karakter) típus létrehozását láttam a legjobb ötletnek, mert rengeteg különböző módon fel lehet őket használni.

Egyenlőre csak bájital árusításra szolgáló merchant van a játékban (élet visszatöltő, energiát növelő, sebzést növelő bájitalok árusítására szolgál). Ahogyan a Questgivernél, itt is a távolságvizsgálat alapján és egy gomb lenyomására történik a vásárlásra szolgáló ablak megjelenítése (Lásd 4.13 ábra). A vásárlás során a játékos karakterének az egyenlege/arany mennyisége csökken, és a vásárolt tárgyak a hátizsákjába kerülnek.

Azt, hogy milyen tárgyakat adhat el egy bizonyos merchant, azt a Settings osztályból egy dictionary-ben tárolt azonosítók beolvasása befolyásolja. Illetve a Settings osztály tartalmaz egy Items dictionary-t is, amelyben a tárgyakra vonatkozó adatokat tárolom. Ilyen adatok például a tárgy neve, típusa, ára, hatása és annak mértéke (például mennyi plusz erő-t ad a tárgy) és időtartama, és a grafikának az elérési útvonala.

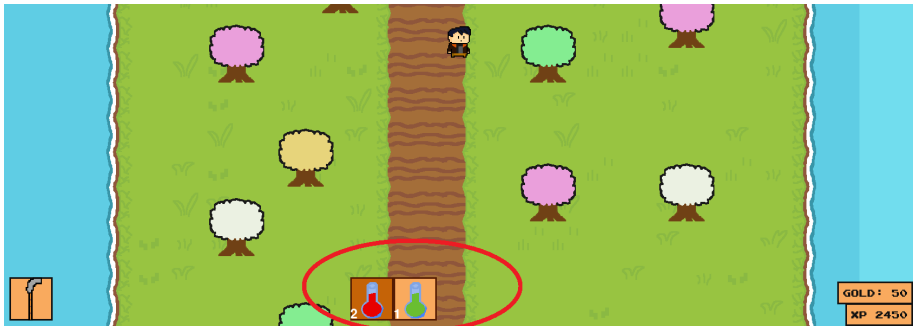


4.13. ábra: Kereskedő karakter

4.4.14. Táska rendszer

A Quickslot azaz táska rendszer, fontos szerepet kap a játék folyamán, mert a merchanttól megvásárolt tárgyak ebbe a táskába kerülnek bele. A játékos ezeket tudja felhasználni amikor szükségét érzi, akár harc közben is.

Öt szabad férőhellyel rendelkezik a táska, ezért jól meg kell válogatni, melyek azok a fontos tárgyak, amelyeket oda szeretne helyezni a játékos. Első nekifutásra egy futószalag szerű működést képzeltem el a táska rendszernek, amelyet azért gondoltam jó ötletnek, mert a megvalósítása egyszerű volt, a megszerzett tárgyak időrendben kerültek bele a táskába. Viszont felmerült a probléma, hogy ha más sorrendben esik kézre a felhasználónak, vagy nincs helye, akkor mit tud tenni. Ezért úgy döntöttem, hogy jobb megoldás lenne, ha az azonos tárgyakkat össze lehetne húzni egy férőhelyre, ezáltal kényelmesebbé válna a használata a táskának. Megtartottam a futószalag szerű megoldást, viszont kiegészült a tárgy összevonással, amelyet egy számláló bevezetésével oldottam meg, amely jelzi a felhasználónak, hogy még mennyi található nála abból az adott tárgy típusból. (Lásd 4.14 ábra)



4.14. ábra: Táska rendszer

4.5. Adatbázis használata a programban

Az online mentés kezeléséhez, illetve a játékban való regisztrációhoz és bejelentkezéshez szükség volt egy backend megvalósítására, azaz egy alkalmazásra, amely kommunikációt végez a MySQL [30] adatbázissal. Az adatbázis struktúrát a MySQL Workbench [31] asztali alkalmazásban építettem fel, eddig az általam használt alkalmazások küzül ez volt, ami felhasználóbarát módon volt megvalósítva. A kommunikációt a FastAPI [29] nevű Python könyvtár segítségével valósítottam meg. A FastAPI egy modern, gyors, könnyű keretrendszer, a microservice-ekhez talán a legjobb választás Python programozási nyelv esetében.

Az adatokat különböző requestek és a megfelelő endpointok segítségével tudjuk elérni, vagy adott esetben felvinni, frissíteni. A FastAPI érzékeli, hogy egy kérés érkezett a rendszerbe, és ha az adott kérés teljesíthető, akkor az esetemben megkezdi a kommunikációt a MySQL adatbázissal és a visszakapott adatokat BaseModell [32] típusú objektumokba tölti, amelyek a Pydantic [33] könyvtárnak köszönhetőek. Ez a modell azért fontos, mert validálja, hogy a kérésben szereplő adatok megfelelnek-e a megadott adatstruktúrának. Miután a validálás is sikeresen lezajlott, a FastAPI visszaküldi a végpont kérésre a megfelelő választ.

5. fejezet

Továbbfejlesztési lehetőségek

A megvalósított játékmechanikák és elemek után fontosnak tartom, hogy megemlítsem az olyan aspektusait a játéknak, amelyeket a jövőben érdemes lenne továbbfejleszteni, illetve olyan funkciókat, amelyeket érdemes lenne még hozzáadni a játékhoz.

5.1. Táska rendszer

A táska rendszer jelenleg elég helyet ad a játékosoknak, hogy tudják kezelni a játék során megvásárolt vagy felvett tárgyakat. Azonban ahogyan a játék fejlesztése fog haladni előre, elkerülhetetlen lesz, hogy több tárgy kerüljön a játékba, így a játékosoknak több helyre lesz szükségük. Ezért érdemes lenne a jelenlegi táska rendszert továbbfejleszteni, hogy több helyet biztosítson a játékosoknak, illetve szeretnék egy funkciót, amellyel a táskában lévő tárgyakat lehessen sorba rendezni, hogy könnyebben megtalálják a kívánt tárgyat.

5.2. Pálya generálás

Jelenleg a világ teljes mértékben statikus, azaz minden pályaelem előre meghatározott helyen van. Ezért érdemes lenne egy olyan funkciót hozzáadni a játékhoz, amely biztosítani tudja a véletlen elemeket a játékban.

Ilyen funkció lehet többek között, az entitások nem csak koorditáta alapján való elhelyezése, hanem egy zóna rendszer létrehozása, amely zóna méreteinek és helyzetének ismeretében, azon belülre véletlenszerű pozícióra rajzolja ki az entitásokat.

Másik szerintem hasonlóan fontos új funkció lenne, a véletlenszerű barlangok létrehozása. Jelenleg egy barlang található a játékban, ahol a főellenség található a legvégén. Viszont jó lenne bővíteni, hogy több kisebb barlang is legyen a játékban, amelyekben kisebb ellenfelek találhatóak, illetve lehetőséget biztosítson kincsek megszerzésére, akár több fegyverre, több arany pénzre, itt is csak a képzelet tud határt szabni.

5.3. Ellenségek

Jelenleg minden ellenségnek ugyanazok a képességei, azonban érdemes lenne egyedi képességeket adni nekik, hogy ne csak a kinézetükben legyenek különbözőek. Jelenleg egy kivétel található, a varázsló, amely tud a varázspálcájával egy varázsgömböt lőni a játékos felé. Viszont ezt a mechanikát is érdemes lenne teljes mértékben átdolgozni.

Ennek a vonzataként következne, hogy az osztályok struktúráját is át kellene gondolni, hiszen most csak egy ellenség osztály található, amely különböző grafikával példányosítja gyakorlatilag ugyanazt a szörnyet. Ezért érdemes lenne szörny típusonként külön osztályt definiálni, és ezután következhet majd az egyedi képességek megvalósítása.

5.4. Felhasználói fiókok biztonságosabbá tétele

A játék jelenlegi állapotában a felhasználók jelszavait titkosítva tárolja az adatbázisban, azonban érdemes lenne továbbfejleszteni a bejelentkezés után történő műveleteket. A megvalósítás során nem használtam a token alapú autentikációt, amely egy sokkal biztonságosabb megoldás lenne az adatok védelmére, hanem ha sikeresen bejelentkezett a játékos, akkor a karakterének adataival történik a későbbiekben a beazonosítás. Ez azért rossz megközelítés, mert a JSON struktúra ismeretében akár felül lehet írni bárki mentését vagy törölni azt. A token alapú autentikáció során a felhasználó adatai nem kerülnek tárolásra a szerveren, hanem egy token-t kap, amelyet minden kérésnél elküld a szervernek, így a szerver tudja, hogy a felhasználó be van jelentkezve, és hozzáférhet az adott erőforráshoz, ezzel ellehetetlenítve az illetékteleneknek a hozzáférést.

6. fejezet

Összefoglalás

A szakdolgozatom célja egy 2 dimenziós akció-kaland játék fejlesztése volt a Pygame könyvtár használatával.

A dolgozat elkészítése során részletesen megismerkedtem a Python programozási nyelvvel, a Pygame játékfejlesztő könyvtárral, illetve a modern és gyors FastAPI web frameworkkel, amely segítségével egy autentikációs rendszert, online játékállapotmentést is készítettem a játékhoz. Nagyon jó döntésnek tartom, hogy a játék fejlesztéséhez az előbb felsorolt eszközöket használhattam, és biztos vagyok benne, hogy újra ezeket fogom választani a jövőben is, ha hasonló projektbe vágom a fejszémet.

Saját és közeli ismerőseim véleménye alapján, akik kipróbálták a játékot, szerintük is nagyon jól sikerült. Az úgynevezett indie típusú játékok kategóriájába tökéletesen beleillik. Sikerült a mai igényeknek megfelelő letisztult és modern kinézet kialakítása. A játék jelenlegi állapota lehetővé teszi a felhasználónak, hogy felhasználói fiókot hozzon létre, internetes, illetve internet nélküli használatot egyaránt. Továbbá változatos küldetések teljesítését, szörnyekkel és más ellenséges karakterekkel való küzdelmet.

Az alkalmazás további kihívásokat és izgalmas feladatokat tartogat a jövőre nézve. A játék továbbfejlesztésével szeretném a játékélményt még jobbá tenni, és új funkciókat hozzáadni a játékhoz. Ilyen funkció lehet a táska rendszer továbbfejlesztése, a pályagenerálás, illetve az ellenségek képességeinek egyedi megvalósítása, de fontos megemlíteni a grafika megtervezését is, amelynek legalább ugyanannyi a szerepe a felhasználói élményre nézve, mint a különböző játékmechanikáknak.

Abban egészen biztos vagyok, hogy a szakdolgozat elkészítése során szerzett rengeteg hasznos ismeretet, tapasztalatot alkalmazni tudom majd a jövőben is.

7. fejezet

Summary

The aim of my thesis was to develop a 2-dimensional action-adventure game using the Pygame library.

While working on the thesis, I gained detailed knowledge of the Python programming language, the Pygame game development library, as well as the modern and fast FastAPI web framework, through which I created a login system and online game state saving feature for the game. I consider it a very good decision to use the tools mentioned above for game development, and I am confident that I will choose them again in the future if I embark on a similar project.

Based on the opinions of myself and close acquaintances who have tried the game, they also believe that it turned out great. It fits perfectly into the category of so-called indie games. I managed to achieve a sleek and modern design that meets today's standards. The current state of the game allows users to create user accounts for both online and offline use. Additionally, it involves completing various missions and engaging in battles with monsters and other enemies.

The game holds further challenges and exciting tasks for the future. With further development of the game, I aim to enhance the gaming experience and add new features. Such features could include further development of the inventory system, area generation, unique implementation of enemy abilities, and it is important to mention the design of graphics, which plays just as important a role in user experience as various game mechanics.

I am confident that the valuable knowledge and experience gained during the making of this project will be applicable in the future.

Források

[1] Ace Lagoon. *The Game Development World Championship*. URL: https://thegdwc.com/.

[2] Unity Technologies. *Unity Documentation*. URL: https://docs.unity3d.com/.

[3] Unity Technologies. *Unity Website*. URL: https://unity.com/.

[4] Microsoft Corporation. *C# Language Reference*. URL: https://docs.microsoft.com/en-us/dotnet/csharp/.

[5] Alexandre Mutel, Kristyna Hougaardn. *Unity and .NET, what's next?* 2022. URL: https://blog.unity.com/engine-platform/unity-and-net-whats-next.

[6] Microsoft Corporation. *Visual Studio*. URL: https://visualstudio.microsoft.com/.

[7] Oracle Corporation. *Java Documentation*. URL: https://docs.oracle.com/en/java/.

[8] Oracle Corporation. *Java*. URL: https://www.java.com/en/.

[9] Mario Zechner. *libGDX*. URL: https://libgdx.com/.

[10] Oracle Corporation. *JavaFX*. URL: https://openjfx.io//.

[11] Caspian Prince. *Lightweight Java Game Library*. URL: https://www.lwjgl.org/.

[12] John R. Rose. *Java Programming Forums*. URL: https://www.javaprogrammingforums.com/.

[13] The C++ Standards Committee. *C++ Standard Library Reference*. URL: https://en.cppreference.com/w/.

[14] *C++*. URL: https://cplusplus.com/.

[15] Mahmud Kabir. *9 Reasons Why Is Python Good For Game Development Still In 2024*. 2023. URL: https://www.oyolloo.com/is-python-good-for-game-development/.

[16] The Python Software Foundation. *Python*. URL: https://www.python.org/.

[17] Pulumi. *Why is Python So Popular?* 2023. URL: https://www.pulumi.com/why-is-python-so-popular/.

[18] Python Software Foundation. *Pygame*. URL: https://www.pygame.org/.

[19] Massachusetts Institute of Technology. *One Laptop Per Child*. URL: https://laptop.org/.

[20] Studio Pixel. *Cave Story*. 2004. URL: https://www.cavestory.org/.

[21] Team Meat. *Super Meat Boy*. 2010. URL: http://www.supermeatboy.com/.

[22] Number None. *Braid*. 2008. URL: http://braid-game.com/.

[23] Tom Rothamel. *RenPy*. URL: https://www.renpy.org/.

[24] SaaSHub. *RenPy VS Pygame*. 2023. URL: https://www.saashub.com/compare-ren-py-vs-pygame.

[25] Pixel-boy. *Ninja Adventure tileset*. 2021. URL: https://pixel-boy.itch.io/ninja-adventure-asset-pack.

[26] Igara Studio S.A. *Aseprite*. URL: https://www.aseprite.org/.

[27] Thorsten Scheuermann. *Tiled*. URL: https://www.mapeditor.org/.

[28] *Python os library*. URL: https://docs.python.org/3/library/os.html.

[29] *FastAPI*. URL: https://fastapi.tiangolo.com/.

[30] Oracle Corporation. *MySQL*. URL: https://www.mysql.com/.

[31] Oracle Corporation. *MySQL Workbench*. URL: https://www.mysql.com/products/workbench/.

[32] *Pydantic Base Model Documentation*. URL: https://docs.pydantic.dev/latest/api/base_model/.

[33] *Pydantic Documentation*. URL: https://docs.pydantic.dev/latest/.

A mellékelt CD tartalma

A játék Windows operációs rendszeren való indításához, a mellékelt CD-n található `Marooned Sailor.exe` nevű futtatható állományt kell elindítani. Mivel tartalmazza a szükséges könyvtárakat és fájlokat, a játék futtatásához nincs szükség egyéb teendőre.

A dolgozathoz tartozó melléklet a következőket tartalmazza:

```
Marooned Sailor/
├── audio/
├── data/
├── dungeontest/
├── graphics/
├── Marooned Saiolor.exe
├── new_map/
└── saves/
Marooned Sailor code/
├── audio/
├── code/
├── data/
├── dungeontest/
├── graphics/
├── new_map/
└── saves/
Marooned Sailor backend/
szakdolgozat/
├── chapters/
├── cover/
├── images/
└── styles/
dolgozat.pdf
```