

SZAKDOLGOZAT



MISKOLCI EGYETEM

Kétdimenziós játék fejlesztése Pygame keretrendszerrel

Készítette:

Szendrei Gábor

Programtervező informatikus

Témavezető:

Dr. Vadon Viktória

MISKOLC, 2023

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 játékokban 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 játékelmény 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 aláírással igazolom, hogy *Szakedolgozat 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

- | | |
|-------|---------------|
| | |
| dátum | témavezető(k) |

- | | |
|------------------------------|-----------------------------|
| témavezető (dátum, aláírás): | konzulens (dátum, aláírás): |
| | |
| | |
| | |

- | | |
|-------|---------------|
| | |
| dátum | témavezető(k) |

- | | |
|-------|---------------|
| | |
| dátum | témavezető(k) |

- dátum szakfelelős

- 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. Java: népszerű választás játékfejlesztők számára	3
2.2.3. C#: Kiemelt szerepű programozási nyelv a játékfejlesztők körében	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	5
2.4.1. Pygam-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	8
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	16
4.2.1.	Tiled	16
4.2.2.	Tervezés	16
4.3.	Felhasználói interfész és beállítások kezelése	17
4.3.1.	Beállítások	17
4.3.2.	Felhasználói fiók	18
4.3.3.	Főmenü	18
4.3.4.	Új játék létrehozása	18
4.3.5.	Játékon belüli menü	19
4.3.6.	Karakter betöltése	19
4.4.	Játékmenet kezelése	19
4.4.1.	Main metódus	19
4.4.2.	A játék fő osztálya	20
4.4.3.	Pálya kezelése	20
4.4.4.	Entitások	21
4.4.5.	Játékos karaktere	22
4.4.6.	Világ és barlangok	22
4.4.7.	Játékállapot mentése	24
4.4.8.	Nem a játékos által vezértelt karakterek	24
4.4.9.	Ellenségek, szörnyek	24
4.4.10.	Küldetést adó karakterek	25
4.4.11.	Kereskedő karakter	26
4.4.12.	Táska rendszer	27
4.4.13.	Zsákmányolás	28
4.4.14.	Fegyverek	29
4.5.	Adatbázis használata a programban	30
5.	Továbbfejlesztési lehetőségek	31
5.1.	Táska rendszer	31
5.2.	Pálya generálás	31
5.3.	Ellenségek	31
6.	Összefoglalás	33
7.	Summary	34
	Források	35

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 valóság-hű 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 módokban 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 izgalmasabb 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 sztori 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 játék motorok, 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 széles körű 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 valóságghű mozgást és ütközéseket biztosítanak, ami a játékelményt 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. Java: népszerű választás játékfejlesztők számára

A Java[4, 5] 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 [6] vagy a JavaFX [7]. 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), 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 Programming Forums* [8]. Ezen a weboldalon a fejlesztők különböző témákban kérhetnek segítséget, vagy éppen segíthetnek másoknak a problémáik megoldásában. Remek leírások, példák és cikkek is találhatóak rajta.

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

A C# [9] egy modern, objektumorientált programozási nyelv, melyet széles körben alkalmaznak a játékfejlesztés területén, különösen a Unity játékmotorral. [10] 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[11] 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 játékelmény 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.

2.2.4. C++: A régebbi játékok elengedhetetlen nyelve

A C++ [12, 13] 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 kapcsolatot 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 [15] 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 több oka van: [16]

- **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.[14] 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 mélyen 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[17] 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. Pygam-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[19] (2004):** Egy kalandjáték, amelyet Daisuke Amaya készített. A játék nagy sikert aratott a játékosok körében, és azóta is népszerű maradt.
- **Super Meat Boy[20] (2010):** Egy gyors tempójú platformjáték, amelyben a játékos egy húsból készült fiút irányít, aki megpróbál eljutni egy húsból készült lányhoz.
- **Braid[21] (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)[18] számítógépét is.
- **Egyszerűség:** Könnyen megtanulható és használható, és kiválóan alkalmas fiatalabb és idősebb játékfejlesztők számára egyaránt.
- **Modularitás:** Lehetőséget ad arra, hogy a különböző modulokat külön-külön inicializálja és használja, így testreszabhatja a fejlesztést az igényeinek megfelelően.

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, játéktípusok 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 motiválóbba tehetjük az oktatást, míg a kognitív kutatásban lehetőséget nyújtanak az emberi kogníció tanulmányozására. 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[22] 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[23] 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 területekkel rendelkezik, és a választás az alkotók célkitűzéseitől és projektjeiktől 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 rendelkeznenek.

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 játékkal 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 sikeréhez. 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.

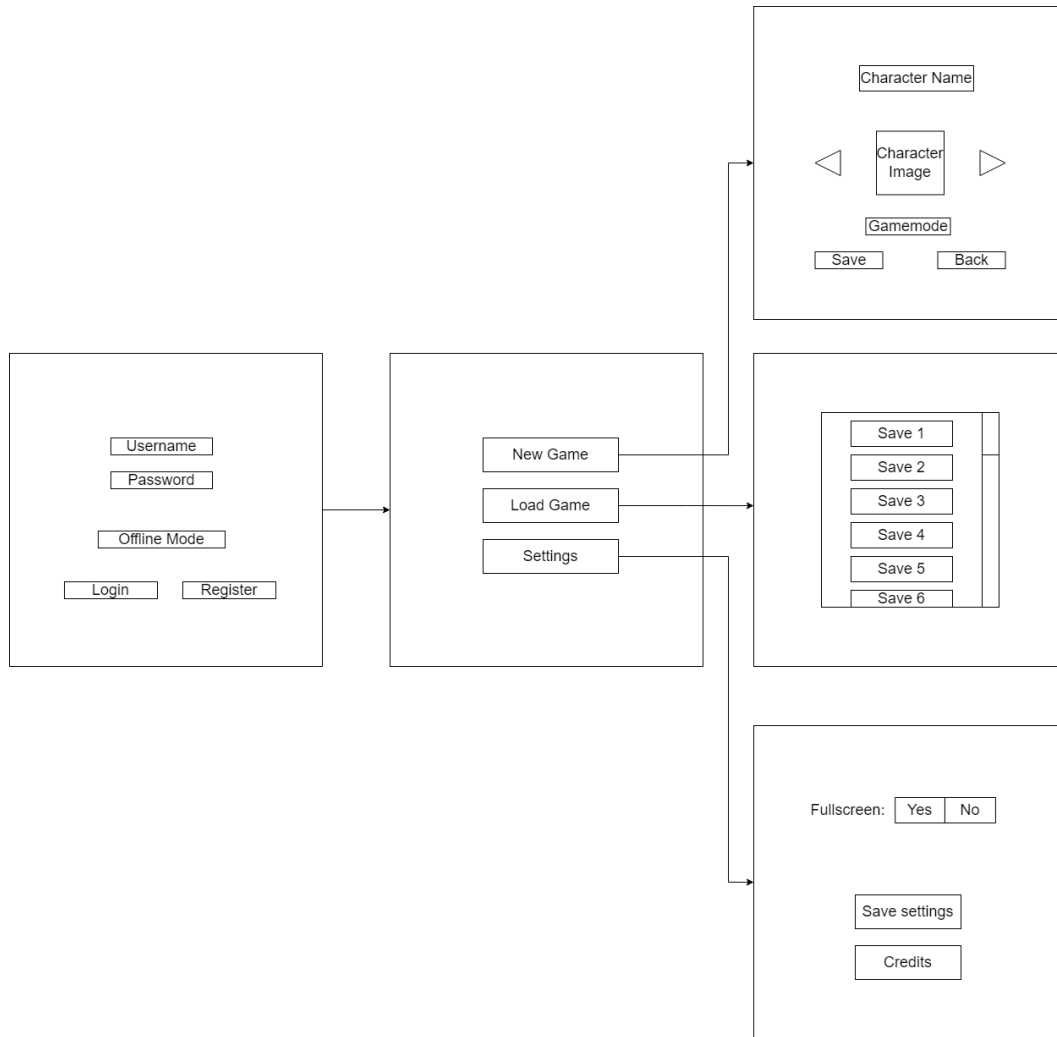
3.2. Grafika

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. (Lásd 3.1 ábra)



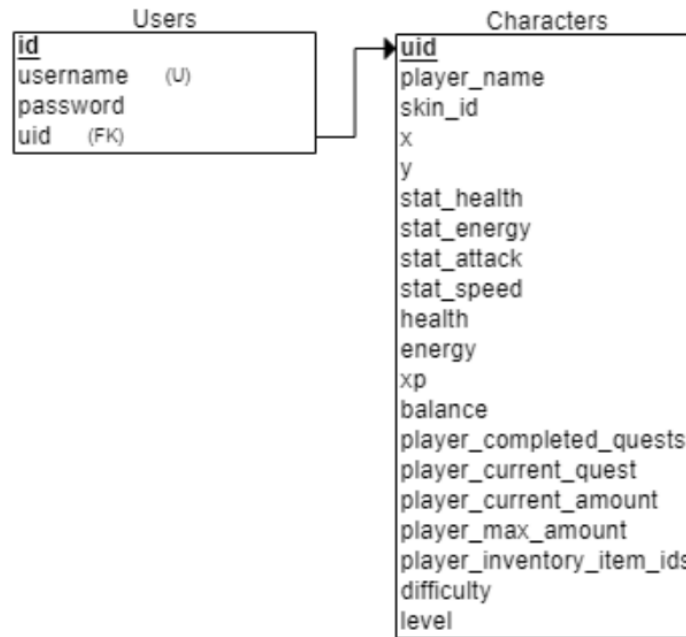
3.1. ábra: Menürendszer terve

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 újratekés után erősödtek. Ehhez a rendszerhez elengedhetetlen volt, hogy meg tudjam különböztetni a játékosokat, illetve, hogy egy adatbázisban tudjam tárolni a játékosok adatait. (Lásd 3.2 ábra)

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 eszemben 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 mentései 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 játékelményt. 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

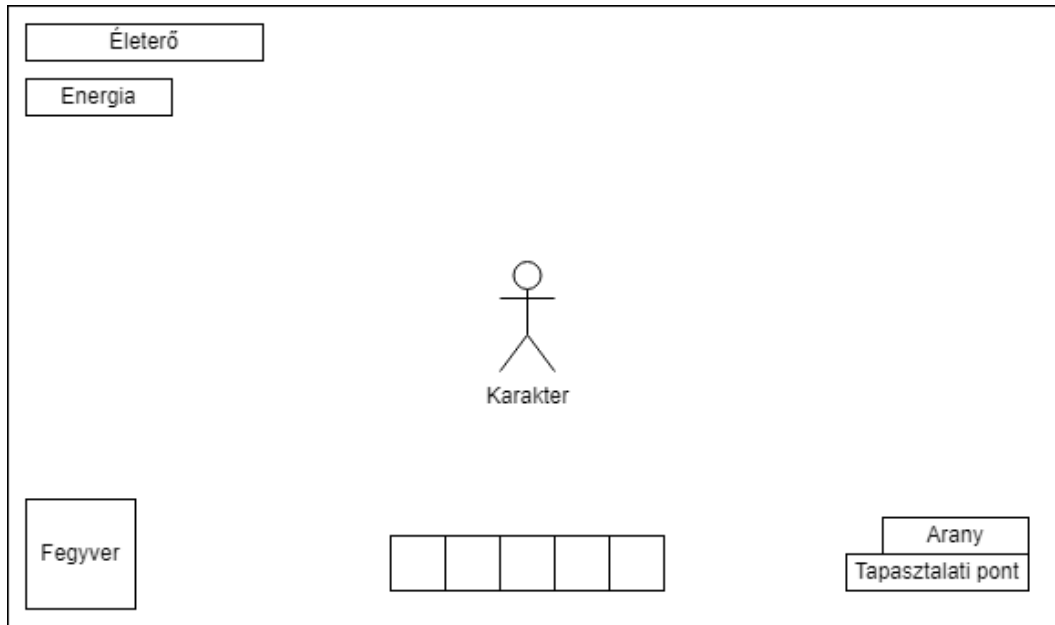
A játékokban fontos a jól megtervezett felhasználói felület amit a játék közben lát a játékos, amely lehetővé teszi a játékosok számára, hogy a lehető legkönnyebben és leggyorsabban elérjék a kívánt funkciókat. Ilyen felület a HUD (Head Up Display), amely segíti a játékosokat abban, hogy valós idejű 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 használt italokat fogja megjeleníteni. 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. (Lásd 3.3 ábra)



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 közelharc fegyver közül, amelyeket a preferenciája 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 szörnyel 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 küldetésekből, és ezáltal fokozatosan erősödjenek a játékban. A lineáris elbeszélés mód 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. Azokat 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 összekapcsolva teremtenek egy gazdag és dinamikus játékélményt az Action RPG (akció-szerepjátékban) játékban, ahol a játékos döntései és cselekedetei hatással vannak a karakter fejlődésére és a játék világára.

4. fejezet

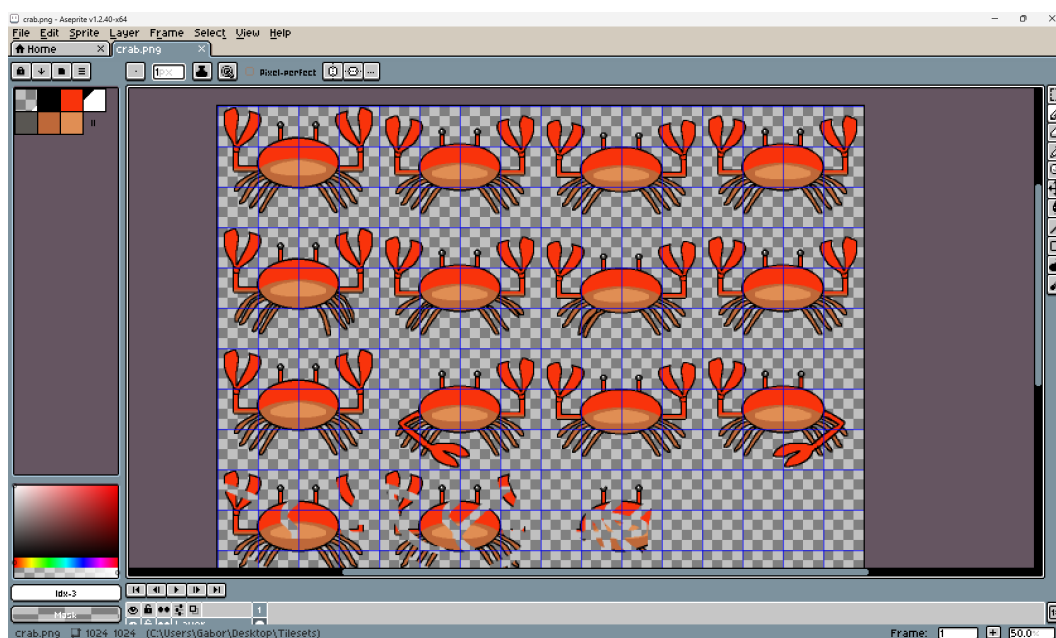
Megvalósítás

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.[24]

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 kiválasztott program az Aseprite volt, ebben készítettem minden játékban megtalálható grafikát. (Lásd 4.1 ábra)



4.1. ábra: Aseprite - Pixelart rajzoló program [25]

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, melyből az összes pálya elemét saját kezűleg rajzoltam meg. A karakterek és környezet rajzainak elkészítésénél pedig meglévő rajzokat használtam inspirációként, és ezeket egészítettem ki saját kreatív ötleteimmel.

Pálya

A pályaelemek létrehozásánál kiemelkedően fontos volt figyelnem arra, hogy a csempek harmonikusan illeszkedjenek egymáshoz, ezáltal kellemes és összefüggő látványt teremtvé.

Entitások

Karakterek és szörnyek megtervezése annyiban tért el, hogy ott figyelni kellett az animációra is, mert élethű mozgást szerettem volna utánózni. 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.

4.2. Pálya megtervezése

4.2.1. Tiled

A pálya megtervezéséhez a Tiled-et 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 számos funkciót kínál, mint például csempekészletek használata, rétegek kezelése, objektumok és kollíziók definiálása, valamint az exportálás különböző formátumokban, például CSV vagy TMX (Tiled Map XML).

Az egyszerű és intuitív felhasználói felülete miatt a Tiled ideális eszköz a játékfejlesztők számára, akik részletes és 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.

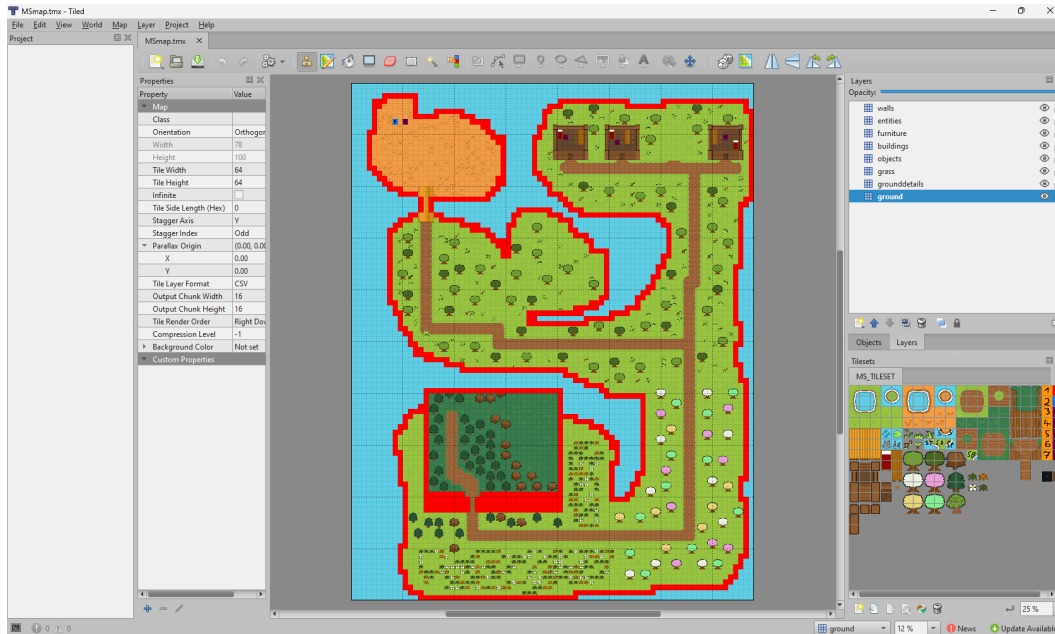
4.2.2. Tervezés

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 4.2 ábra piroskörvonal), ezeket a rétegeket 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. (Lásd 4.2 ábra)



4.2. ábra: Tiled - Pálya szerkesztő [26]

4.3. Felhasználói interfész 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 játéklablak méretét, a hangerejét és egyebeket. Az osztály segítségével ezeket a beállításokat be lehet tölteni és menteni egy JSON fájlba.

Az osztályban található statikus osztályváltozók az alapértelmezett beállításokat tárolják, és a program különböző részeiben felhasználhatók. Az osztályban található statikus osztályváltozók az alapértelmezett értékeket tárolják, és a program különböző részeiben felhasználhatók, anélkül hogy az osztály példányt kellene létrehozni belőle, így könnyen hozzáférhetőek.

Az osztály további adatokat is tartalmaz, például hangokat, grafikákat, karaktereket, ellenségeket, lövedékeket és fegyvereket. Mindezek az adatok részletezik a játékbeli objektumok tulajdonságait.

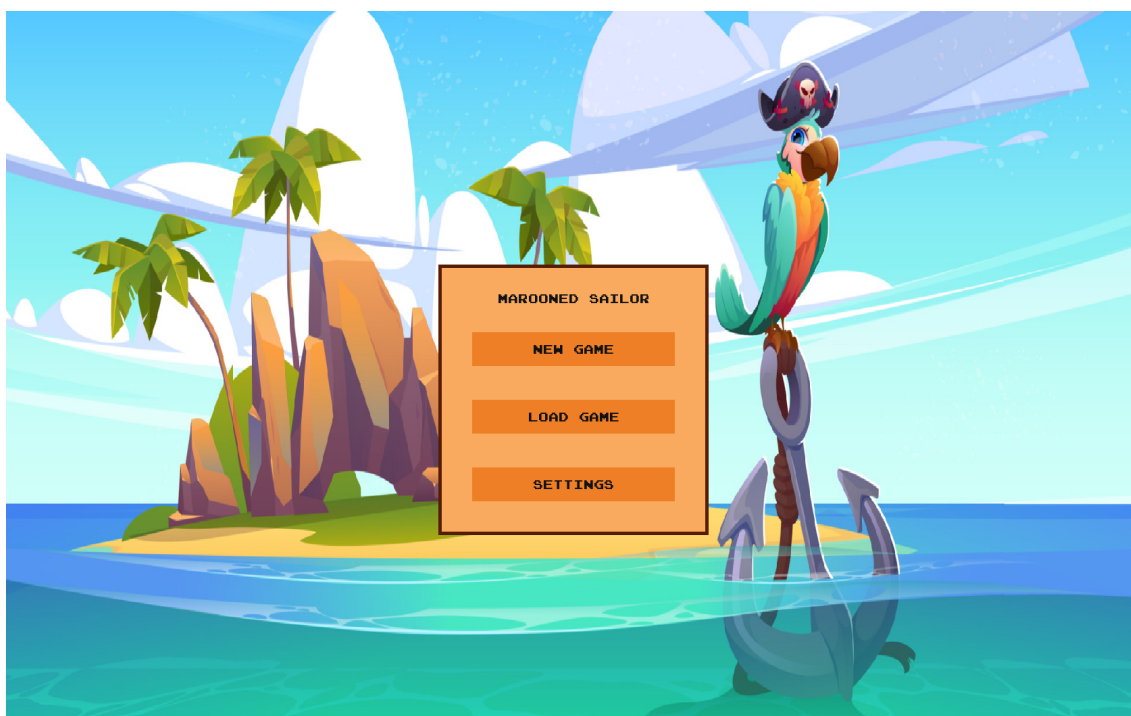
Az osztály emellett definiálja a felhasználói felület színeit és stílusait, amelyek a játék megjelenítéséhez használatosak. Ezek a színek és stílusok segítenek a felhasználói felület személyre szabásában.

4.3.2. Felhasználói fiók

Ez rendszer kezeli a regisztrációhoz és a bejelentkezéshez kapcsolódó folyamatokat. Ilyen folyamat többek között a felhasználói felület megjelenítése, de a tartalom ellenőrzés és hibakezelés is.

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 három menüpont található meg, az új játék létrehozása, meglévő mentés betöltése, illetve a beállítások. (Lásd 4.3 ábra)



4.3. ábra: Főmenü

4.3.4. Új játék létrehozása

Az új játék menüpont kiválasztása után a játékosnak meg kell adni a karakterének a nevét, illetve választhat különböző karakter kinézet 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" módot választja, akkor három funkcióval bővül a játékmenet. Az első ilyen funkció, hogy ha elfogy a játékos életeréje, 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 újakezdődik, amely azt eredményezi, hogy a szörnyek erősödnek, és nagyobb kihívás lesz újra végigvinni az összes küldetést. Illetve tartalmaz egy ranglista menüpontot, amely jelzi a legügyesebb játékosok hányszor tudták a "challenge" mód használatával végigvinni az összes küldetést. (Lásd 4.4 ábra)



4.4. ábra: Új játék létrehozása

4.3.5. Játékon belüli menü

Játékmenet közbeni menü amely lehetővé teszi a játékállapot mentését kezelő metódus elindítását a Save gomb megnyomásával és egyben egy pillanat állj funkció elérését is.

A pillanat állj funkciót úgy valósítottam meg, hogy az összes játékbeli sprite group, azaz felületi csoport 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 gomb megnyomásával tehet meg a játékos, onnan folytatódnak az események ahol abbamaradtak.

4.3.6. Karakter betöltése

A játékosnak lehetősége van a korábban elmentett játékállás betöltésére is, amennyiben van ilyen. A betöltés után a játék folytatódik ahol abbahagyta. Egy lista elrendezésben látja a felhasználó a korábbi mentéseit. Előre rendezi az offline mentéseket a listában, és csak utánuk tölti be az online mentéseket. Egy kattintás után már be is tölti az adott játék állást.

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, és ennek a főciklusnak a futása irányítja a játékmenetet.

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

Amennyiben a menürendszer visszaadja a `menuenums.GAME` értéket, az azt jelenti, hogy a játékot választottuk a menüből, és belépünk a játék főciklusába. Itt történik minden, a játékmenet során szükséges frissítés, eseménykezelés és kirajzolás. Itt szabjuk meg a képfrissítés mértékét is a `'self.clock.tick(Settings.FPS)'` sorral, amely a másodpercenként kirajzolandó képkockák számát határozza meg.

Ez az osztály tehát létfontosságú szerepet játszik abban, hogy a játék működése zökkenőmentes legyen, és a kódban látható példa segítségével könnyen megérthető, hogyan történik mindez. (Lásd 4.1 programkód)

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. A játék fő osztálya

Nagyon fontos szerepet tölt be a játék futása során, ez az osztály kezeli a mentéseket erről a következő szekcióban lesz szó bővebben. Továbbá a felhasználói felület kezelése és használata is itt történik. Illetve az összes játékon belüli menü funkció szintén itt példányosodik és van kezelve. Ilyen menük a játékon belüli menü, ami a pillanat állj funkcióval együtt működik, a ranglista, karakter fejlesztési menü, és még az irányítások felület is, ahol a felhasználó meg tudja nézni, melyik gomb mire való.

4.4.3. Pálya kezelése

Ez a másik nagyon lényeges funkció, ez a `GameHandler`-ben kerül példányosításra. Feladata a világok, entitások létrehozása és kezelése. Emellett a kamera kezelés is ide tartozik. Ezeket a különböző felületeket `Sprite Group`-okba szervezve kezeli, így könnyen tudja őket elérni és frissíteni. Legfontosabb ezek a `sprite` csoportok közül talán a `Visible Sprite` csoport, amely a játékos által látható felületeket tartalmazza. A kamera is ezeket rendereli ki úgy, hogy ezeket a csempéket `y` tengely szerint rendezi és úgy jeleníti meg, ezáltal egy hamis 3D hatást keltve. Megtalálható még `obstacle`,

attack, és attackable felület csoportok is, amelyek a játékmenet során fontos szerepet játszanak, különböző mechanikák végrehajtásánál.

4.4.4. Entitások

Ez az osztály egy szülő osztály, ami 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 másik egy Sin függvény alapján ad vissza 0-255 értéket, ez később a kapott sebzés kijelzésére szolgál. A harmadik metódus az ütközés érzékelésre, kezelésére szolgál.

Az ütközés érzékelés lényegében úgy működik, hogy nézzük az entitás mozgási irányát függőlegesen, vagy vízszintesen közlekedik, és az adott iránytól függően vizsgáljuk, hogy az entitás hitbox-a ütközik-e valamelyik akadállyal. Ha igen, akkor az entitás irány szerinti hitbox-át az akadály ellenkező irányú hitbox-ához igazítjuk, így megakadályozva az áthaladást. (Lásd 4.2 programkód)

Programkód 4.2. Ütközés kezelés

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

A mozgást a move függvény segítségével hajtja végre, amelynek a lényege, hogy az entitás rendelkezik egy irány változóval, ami tárolja, hogy merre néz ez lehet (fel, le, jobbra, balra) és ezeknek az irányoknak a normalizálásával és a paraméterként megkapott sebesség értékkel meghatározza a következő pozícióját az entitásnak. Ez a pozíció módosítás a 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 és nem a kép a hitboxhoz. Ez azért fontos, mert így megelőzhető az olyan hibák előfordulása, amit a hitbox és a kép közötti eltérés okozhatna.

4.4.5. 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 is dinamikusan változik, és a játékos lehetőséget kap a fegyver cseréjére a játék során.

A karakternek vannak életpontjai és energia szintje, amelyek meghatározzák a túlélését a játékban. Amennyiben az életpontja nullára csökken, a karakter meghal, de van lehetőség a visszatérésre vagy újratekérésre 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.

4.4.6. Világ és barlangok

Egy közös szülő osztályból származnak le, ez azért fontos, hogy a különböző világokat könnyen tudjam kezelni és különböző 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 felhívni ott, ahol szükséges ahelyett, hogy minden világnak lenne egy külön példánya. Továbbá ilyen opció a térképmegnyitás, és az azzal kapcsolatos összes funkció, hol található a következő küldést adó nem játékos karakter. mert a játék története jelenleg teljesen lineáris, tehát egy adott sorrendben történhet csak a küldetések teljesítése. Amit még fontos megemlíteni ezzel az osztállyal kapcsolatban, az a szörnyek által dobott tárgyak, ilyen tárgy jelenleg, a tapasztalati pont, és az aranypénz. Ezeket a tárgyakat a játékos fel tudja venni, és a játék során fel tudja használni.

Pálya generáláshoz szükséges adatok

Említettem a korábbi fejezetben, a pálya megtervezését a **Tiled**-el végeztem. (Lásd 4.2.1) Első megközelítésemkor CSV (vesszővel elválasztott értékek) fájl formátumba mentettem ki a tervezőben létrehozott rétegeket. Egy ilyen dokumentum úgy néz ki, hogy ahova nem helyeztem le a tervezőben objektumot ott '-1' érték szerepel, ellenkéntben 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 4.3 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 készítettem python szótárat a csempe azonosító számából, illetve annak a csv mátrixban elhelyezkedő koordinátáiról. (Lásd 4.4 programkód)

Programkód 4.3. 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)

            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}")
```

Programkód 4.4. Minta az átalakított struktúrára

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

Pálya generálás

Ahogy a grafika és a pálya rétegek elkészültek, melyek ki lettek exportálva. A játékban való megjelenítését több lépésben valósítottam meg, először is a képeket be kell tudnom olvasni, ezt az os könyvtár walk függvényének segítségével oldottam meg. Miután sikerült bejárni a mappát, előállította az egyes képekhez vezető útvonalat, ezután a `pygame.image.load` függvényével betöltte a képeket. 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 szótárak segítségével valósítottam meg. Ezután egy ciklusban végig iterálva a szótáron, a képek listájából kiválasztotta a megfelelő képet, és a megfelelő koordinátákat, amelyeket segítségével példányosítom a Tile osztályt, később ennek segítségével végezte el a kirajzolást. Mivel a szótárak között szerepel az entitásokra vonatkozó réteg is, ezért azokat is az előbb említett iterációban kezeltem, és példányosította a hozzájuk tartozó osztályokat azonosító alapján szűrve.

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

Szótár esetében:

Futási idő: 108.64 ms

CSV fájl esetében:

Futási idő: 879.92 ms

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

4.4.7. 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 keresztül kommunikálva egy mysql adatbázisban, ha online módot választott a felhasználó.

4.4.8. Nem a játékos által vezértelt karakterek

Nem játékos által vezértelt karakterek azaz NPC-k egy kalandjáték során elengedhetetlen részei a történetnek és a játékmenetnek. Ahogy az összes többi szörny, játékos osztály is az Entity osztályból származik le, mert adott esetben szükség lehet mozgásra 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.

4.4.9. Ellenségek, szörnyek

Az ellenségek a játékban a küldetések után talán a legfontosabb mechanikai a játéknak, mert a velük való harc kiemelt szerepet kap a történet során. Az ellenségek is egy közös szülő osztályból származnak le, ez azért volt fontos, hogy a különböző

ellenségeknek különböző tulajdonságokat tudjak adni. Például a sebesség, életerő, támadási sebesség, támadások hangjai. Ezeket a tulajdonságokat settings statikus osztály `monster_data` dictionary-ben tárolom és olvasom be.

A szörnyek mozgását úgy valósítottam meg, hogy ha a játékos egy adott sugaron belül ér a szörnyhöz képest, ez az úgynevezett érzékelési sugár, akkor kiszámolom vektorokkal a játékos távolságát, és ha a távolság nagyobb, mint 0, azaz nem egy helyen állnak, akkor meghatározom az irányt a két vektor különbségének normalizálásával. A távolság érték azért fontos, hogy megállítsa mikor ér a játékos karaktere a szörnyhöz támadási távolságon belülre. Az irány érték pedig a szörny mozgatásához szükséges.

Három különböző státusszal rendelkezhet egy ellenség. Támadási státuszt kap, ha támadási sugaron belül tartózkodik a játékos karaktere, és ha egy bizonyos idő eltelt az előző támadás óta. Mozgás státuszt kap, ha a játékos karaktere a szörny érzékelési sugarán belül van, de nem támadási sugaron belül. Ha a játékos karaktere nincs az érzékelési sugarán belül, akkor pedig nyugalmi státuszt kap. Ezek a státuszok határozzák meg a szörnyek viselkedését és a megjelenített grafikát is a játék során.

4.4.10. Küldetést adó karakterek

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

A Questgiver NPC típust a World osztály példányosítja és a példányok osztályváltozóinak inicializálásakor olvassa be a saját id-jéhez tartozó küldetéseket a Settings osztály `npc_data` dictionary-jéből. Majd ezt követően a World osztály létrehozza a Player osztályból is egy példányt, amely egészen a LevelHandler osztálytól kap paraméterként egy `questgivers_quest_setup` függvényt, amely egy trigger után vizsgálja hogy a játékos álltal már teljesített küldetés szerepele még Questgivernél és, ha igen, akkor kitörli azt. Ezáltal elkerülhető a küldetés ismétlődése. Küldetést nem lehet kihagyni, mindegyiket sorban kell teljesíteni, ahogy a történet halad előre.

Az questgiver típusú NPC-k képesek egy előre megírt dialógus megjelenítésére (Lásd 4.5 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.



4.5. ábra: Dialógus

4.4.11. Kereskedő karakter

Egy kalandjátékban az előrehaladás mellett fontos, hogy a nehezen megszerzett tárgyakkal, esetünkben az arannyal lehessen mit kezdeni, ezért egy Merchant NPC 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). A továbbfejlesztési lehetőségek bekezdésben jobban kifejtem, mikre lehet még használni a merchant típusú NPC-t. Ahogyan a Questgivernél, itt is a távolságvizsgálat alapján és egy gomb lenyomására történik a vásárlásra szolgáló ablak megjelenítése (Lásd 4.6 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 dictionaryben tárolt id-k 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 mennyisége (Például mennyi plusz erő-t ad a tárgy), illetve a időtartamra vonatkozó megkötést és talán a legfontosabbat a grafikának az elérési útvonalát.



4.6. ábra: Kereskedő karakter

4.4.12. Táska rendszer

Az Inventory azaz táska rendszer, fontos szerepet kap a játék folyamán, mert a merchanttól megvásárolt tárgyak ebbe a hátizsákba kerülnek bele és a játékos ezeket tudja felhasználni amikor szükségét érzi.

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, mert megszerzett tárgyak időrendben kerültek bele a táskába. Viszont felvetette a problémát, 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 tárgyakat 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 tárgyból. (Lásd 4.7 ábra)



4.7. ábra: Táksa rendszer

4.4.13. Zsákmányolás

A zsákmányolás funkciót azért tartottam fontosnak, mert plusz motivációt nyújt a játékosnak legyőzni a szörnyeket, ha szerezhetsz is tőlük tárgyakat. Egyenlőre a szörnyek három különböző tárgy dobására képesek. Tapasztalati pont bogyókat hagynak hátra, amelyek előre meghatározott mennyiségű tapasztalati pontot tartalmaznak, és mindegyik szörny ilyet hagy maga után a legyőzése után. Ezenkívül aranyat is dobhatnak, 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 példányosításakor paraméterként megkapják és, ha elfogy az életpontjuk akkor hívják meg azt. Ez a módszer meghatározza a szörny legutóbbi pozíciójához viszonyítva egy 100x100-as négyzeten belüli pozíciót, ahova a zsákmányolható tárgyak kerülni fognak. Ezt követően a szörny típusához viszonyítva kiolvassa a `Settings` osztály `monster_data` dictionary-jéből, hogy az adott szörny milyen tárgyakat dobhat, majd ez alapján az előre meghatározott esély alapján véletlenszám generálásával eldönti, hogy milyen tárgyakat hoz létre. A létrehozás úgy történik, hogy az adott pálya példánynak a `loots` osztályváltozójához hozzá fűz egy `Loot` osztály példányát amely tárgy nevét, mértékét, és helyét.

A kirajzolása a felvehető tárgyaknak a `visible_sprite` csoport frissítésével történik, mert a `Loots` osztály ahogy a többi objektum is, a `pygame.sprite.Sprite` osztályból származik le.

Tárgyak felvétele a tárgy pozíciója és a játékos karakternéket távolságát vizsgálva történik, ha elég közel megy a játékos a tárgyhoz, akkor kitörli a `loots` listából a feltett tárgyat és az értékeit hozzáadja a játékos karakterének a tulajdonságaiban tárolt értékekhez. (Lásd 4.8 ábra)



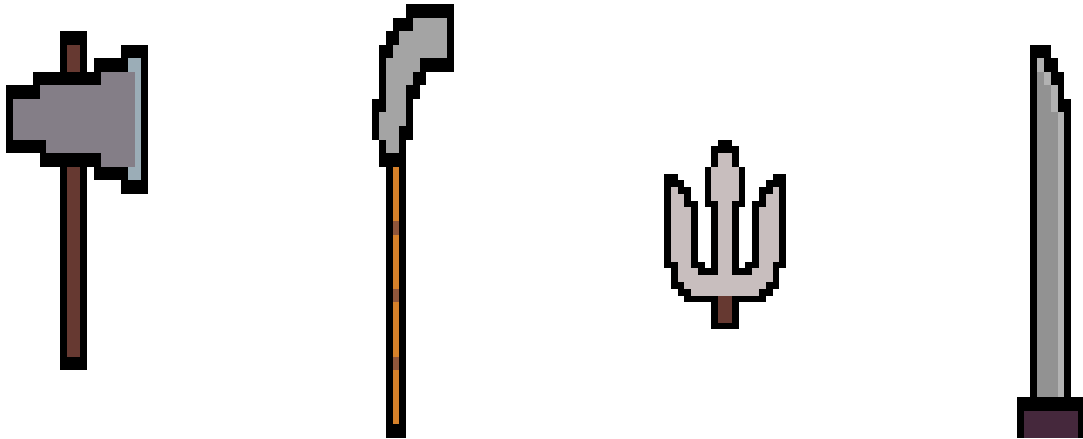
4.8. ábra: Zsákmányolás funkció

4.4.14. Fegyverek

A különböző fegyverek használata szintén egy elengedhetetlen része lett a játéknak. A fegyvereket a játékos karaktere használja a szörnyek elleni harc során.

Ezen harcolásra szolgáló eszközök kirajzolása a támadás gomb lenyomása után történik a Weapon osztály példányosításával. Ez az osztály a játékos karaktere által kezében tartott fegyver 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 felületet. A csapás animációt mind a négy irányban külön-külön kellett megtervezni, mert a fegyverek nem egyformán néznek ki minden irányból. Sinus függvény segítségével valósítottam meg a megjelenített fegyver forgatását egy amplitúdó értékkel, amely a fegyver típusától függően változhat.

Ahogy a többi látható objektum, ezek a fegyverek is a visible sprite csoport része, viszont rendelkezik az attack_sprites csoporttal is, amely segít megkülönböztetni a többi felülettől a fegyvereket, ezáltal támogatva a támadási logika megvalósítását. A támadási logika egy metódus, amely ezeken az attack_sprites csoporton végig iterál, és vizsgálja az attackable_sprites felületekkel való ütközést, amelyeket olyan entitások kaphatnak, amelyekkel a játékos meg tud küzdeni a játék során, legyen az egy szörny vagy akár egy kiüthető fűcsomó. (Lásd 4.9 ábra)



4.9. ábra: A játékban megtalálható fegyverek

s

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 az adatbázissal. Ezt a FastAPI 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 URL 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 esetben megkezdődik a kommunikáció a mysql adatbázissal és a visszakapott adatokat BaseModel típusú objektumokba tölti, amelyeket a Pydantic könyvtár segítségével tudok kezelni. 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 játék mechanikák és elemek után fontosnak tartom, hogy a jövőre tekintettel megemlítssem az olyan aspektusait a játéknak, amelyeket a jövőben érdemes lenne továbbfejleszteni, illetve olyan funkciókat, amelyeket érdemes lenne még hozzáadni a játékhoz.

5.1. Táskarendszer

A táskarendszer 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 táskarendszert 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 koordináta alapján való elhelyezése, hanem egy zónarendszer 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 jobb fegyverre, több arany pénzre, itt is csak a képzelet tud határt szabni.

5.3. Ellenségek

Jelenleg minden ellenségnek ugyanazok a képességei vannak, azonban érdemes lenne egyedi képességeket adni nekik, hogy ne csak a kinézetükben legyenek különbözőek, hanem a képességeikben is. 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 ugyan azt a szörnyet. Ezért érdemes lenne szörny típusonként külön osztályt definiálni, és ezután következhet majd az egyedi képességek megvalósítása.

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 bejelentkezési 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 megfelelően letisztult és modern kinézet kialakítás megvalósítása. A játék jelenlegi állapota lehetővé teszi a felhasználónak, hogy felhasználói fiókot hozzon létre, internet 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 játékelményt még jobbá tenni, és új funkciókat hozzáadni a játékhoz. Ilyen funkció lehet a táska rendszer továbbfejlesztése, a pálya generálás, illetve az ellenségek képességeinek egyedi megvalósítása, de fontos megemlíteni a grafika megtervezését is, amelynek legalább ugyanannyi szerepe a felhasználói élményre nézve van mint a különböző játék mechanikáknak.

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.

During the preparation of 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 my own and close acquaintances who have tried the game, they also believe that it was very successful. It fits perfectly into the category of so-called indie games. I managed to achieve a sleek and modern design that meets today's demands. 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 hostile characters.

The application 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, level 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 preparation of the thesis 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] Oracle Corporation. *Java Documentation*. URL: <https://docs.oracle.com/en/java/>.
- [5] Oracle Corporation. *Java*. URL: <https://www.java.com/en/>.
- [6] Mario Zechner. *libGDX*. URL: <https://libgdx.com/>.
- [7] Oracle Corporation. *JavaFX*. URL: <https://openjfx.io/>.
- [8] John R. Rose. *Java Programming Forums*. URL: <https://www.javaprogrammingforums.com/>. ■
- [9] Microsoft Corporation. *C# Language Reference*. URL: <https://docs.microsoft.com/en-us/dotnet/csharp/>. ■
- [10] Alexandre Mutel, Kristyna Hougaardn. *Unity and .NET, what's next?* 2022. URL: <https://blog.unity.com/engine-platform/unity-and-net-whats-next>.
- [11] Microsoft Corporation. *Visual Studio*. URL: <https://visualstudio.microsoft.com/>.
- [12] The C++ Standards Committee. *C++ Standard Library Reference*. URL: <https://en.cppreference.com/w/>.
- [13] C++. URL: <https://cplusplus.com/>.
- [14] 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/>.
- [15] The Python Software Foundation. *Python*. URL: <https://www.python.org/>.
- [16] Pulum. *Why is Python So Popular?* 2023. URL: <https://www.pulum.com/why-is-python-so-popular/>.
- [17] Python Software Foundation. *Pygame*. URL: <https://www.pygame.org/>.
- [18] Massachusetts Institute of Technology. *One Laptop Per Child*. URL: <https://laptop.org/>.
- [19] Studio Pixel. *Cave Story*. 2004. URL: <https://www.cavestory.org/>.
- [20] Team Meat. *Super Meat Boy*. 2010. URL: <http://www.supermeatboy.com/>.
- [21] Number None. *Braid*. 2008. URL: <http://braid-game.com/>.

- [22] Tom Rothamel. *RenPy*. URL: <https://www.renpy.org/>.
- [23] SaaSHub. *RenPy VS Pygame*. 2023. URL: <https://www.saashub.com/compare-ren-py-vs-pygame>.
- [24] Pixel-boy. *Ninja Adventure tileset*. 2021. URL: <https://pixel-boy.itch.io/ninja-adventure-asset-pack>.
- [25] Igará Studio S.A. *Aseprite*. URL: <https://www.aseprite.org/>.
- [26] Thorsten Scheuermann. *Tiled*. URL: <https://www.mapeditor.org/>.

CD Használati útmutató

Ennek a címe lehet például *A mellékelt CD tartalma* vagy *Adathordozó használati útmutató* is.

Ez jellemzően csak egy fél-egy oldalas leírás. Arra szolgál, hogy ha valaki kézhez kapja a szakdolgozathoz tartozó CD-t, akkor tudja, hogy mi hol van rajta. Jellemzően elég csak felsorolni, hogy milyen jegyzékek vannak, és azokban mi található. Az elkészített programok telepítéséhez, futtatásához tartozó instrukciók kerülhetnek ide.

A CD lemezre mindenképpen rá kell tenni

- a dolgozatot egy `dolgozat.pdf` fájl formájában,
- a LaTeX forráskódját a dolgozatnak,
- az elkészített programot, fontosabb futási eredményeket (például ha kép a kimenet),
- egy útmutatót a CD használatához (ami lehet ez a fejezet külön PDF-be vagy Markdown fájlként kimentve).