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 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. 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. Ez az oldal arra szolgál, hogy bemutassa a számítógépes játékok világát, és megmutassa, hogy milyen élményeket nyújtanak.

A számítógépes játékok lehetővé teszik számunkra, hogy elmerüljünk olyan világokban, amelyekben különböző fantáziavilágokban kalandozhatunk, izgalmas cselekmények részesei lehetünk, és versenyezhetünk a világ minden tájáról érkező játékosokkal. 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, mindegyiknek saját kihívásai és élményei vannak.

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 a világ minden tájáról érkező emberekkel.

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 nézzük meg, 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.

Kreativitás: A jó játékok egyediséget és kreativitást követelnek meg a játéktervezőktől. Az új és izgalmas játékmechanizmusok kitalálása kulcsfontosságú.

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 rejlenek:

Kreatív kibontakozás: A játékfejlesztés lehetőséget ad a kreativitás megnyilvánulására, ahol csak a képzelet szab határt.

Közösség és verseny: A játékfejlesztők részt vehetnek aktív közösségekben, és akár versenyeken is, ahol megmutathatják tehetségüket és fejlődésüket.

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

[1] Az Unity 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.

Az 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, például Windows, iOS, Android vagy konzolokra.

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

2.2.2. Java: Népszerű választás játékfejlesztők számára

[2] A Java 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 vagy a JavaFX. 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 Java nyelven.

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

[3]

A C# egy modern, objektumorientált programozási nyelv, melyet széles körben alkalmaznak a játékfejlesztés területén, különösen az Unity játékmotorral. Az 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# rendelkezik egy erős és fejlett fejlesztői környezettel, amely segíti a fejlesztőket a játéktervezésben és kódolásban. Az integrált fejlesztői eszközök, például a Visual Studio, hatékony eszközöket nyújtanak a kódírás, hibakeresés és teljesítményoptimalizálás terén.

Az 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.

2.2.4. C++: A régebbi játékok elengedhetetlen nyelve

[4]

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

Az eszközök és nyelvek kiválasztása a projekt specifikus igényektől és a fejlesztői készségektől függ. Fontos megérteni, hogy minden eszköznek és nyelvnek megvannak a saját előnyei és korlátai, és ezeket a projekt céljaival és a fejlesztőcsapat készségeivel kell összeegyeztetni annak érdekében, hogy a lehető legjobb játékélményt nyújtsák a játékosoknak.

2.3. Python a játékfejlesztésben

A Python 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?

1. 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 a fejlesztési időt csökkentheti és csökkentheti a hibák számát.

2. Gyors Fejlesztés: A Python lehetővé teszi a gyors prototípusok létrehozását. A gyors prototípusok segítenek a játék ötleteinek gyors validálásában és tesztelésében, mielőtt hosszú fejlesztési ciklusokba kezdünk.

3. Széles Közösség és Támogatás: A Python 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. 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 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.

2.4. Pygame

A Python programozási nyelv számos lehetőséget kínál játékfejlesztőknek a játékok készítéséhez, és az egyik ilyen lehetőség a "Pygame" nevű keretrendszer. A Pygame egy népszerű és könnyen elsajátítható eszköz a 2D játékok fejlesztéséhez. Ez a keretrendszer számos funkciót és eszközt kínál a grafikai megjelenítés, hangkezelés és inputkezelés terén, így a fejlesztők könnyedén kezelhetik ezeket a fontos aspektusokat.

A Python és a Pygame kombinációja kiválóan alkalmas arra, hogy gyorsan és hatékonyan játékokat hozzunk létre. A Python nyelv egyszerűsége és olvashatósága lehetővé teszi, hogy a fejlesztők a játékmechanizmusokra és a játékélményre összpontosítsanak, anélkül hogy túlzott időt kellene fordítaniuk a kód bonyolultságának kezelésére.

Az erős fejlesztői közösség és a dokumentáció elérhetősége további előnyt jelent a Pygame használatakor. A fejlesztők könnyen hozzáférhetnek támogatáshoz és útmutatáshoz, ami segíti őket a projektjük sikerességében.

Összességében a Pygame egy kiváló eszköz a Python használói számára, akik játékokat szeretnének fejleszteni. Az egyszerűsége és a sokoldalúsága lehetővé teszi a kreatív játékfejlesztést anélkül, hogy a technikai részletekre túlzottan összpontosítanánk.

2.4.1. Pygame lehetőségei

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

Grafikai megjelenítés: A Pygame lehetővé teszi a grafikus elemek létrehozását és kezelését, beleértve a sprite-okat, háttérképeket és rajzolási funkciókat is.

Hangkezelés: A könyvtár lehetővé teszi hangfájlok lejátszását, hanghatások és zene hozzáadását a játékhoz.

Inputkezelés: Pygame segítségével könnyedén kezelhetők a felhasználói inputok, például a billentyűzet és egér események.

Multiplatform támogatás: Pygame rendkívül hordozható, és szinte minden platformon és operációs rendszeren fut, beleértve Linuxot, Windowst, MacOS-t és másokat.

Hordozhatóság: Pygame alkalmazható számos eszközön és operációs rendszeren, beleértve a kézi eszközöket, játékkonzolokat és a One Laptop Per Child (OLPC) számítógépét is.

Egyszerűség: A Pygame könnyen megtanulható és használható, és kiválóan alkalmas fiatalabb és idősebb játékfejlesztők számára egyaránt.

Modularitás: A Pygame lehetőséget ad arra, hogy a különböző modulokat különkülön inicializálja és használja, így testreszabhatja a fejlesztést az igényeinek megfelelően.

2.4.2. Pygamemel Készített Sikeres Játékok

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

- Stardew Valley (2016): Egy életszimulátor, amelyben a játékos egy kisvárosban él és gazdálkodik. A játékot a játékmenetéért, a karaktereiért és a világépítéséért dicsérték.
- Celeste (2018): Egy nehéz platformjáték, amelyben a játékos egy fiatal nőt irányít, aki megpróbál felmászni egy hegyre.
- Super Meat Boy (2010): Egy gyors tempójú platformjáték, amelyben a játékos egy húsból készült fiút irányít, aki megpróbál eljutni egy másik húsból készült lányhoz.
- Undertale (2015): Egy indie szerepjáték, amelyben a játékos egy kisfiút irányít, aki egy földalatti világba esik.
- M̈inecraft(Mojang): Bár a M̈inecrafttegy saját motorral rendelkező játék, a Pygame inspirálta a megjelenésekor, és az alfa verziója Pygame segítségével készült.

A játék eredetileg egy egyszerű blokképítő játék volt, amelyben a játékosok szabadon építhettek és alkothattak. A játék később fejlődött, áttért a Java nyelvre és egy hatalmas, többjátékos sandbox világgá vált.

2.4.3. Prototípusok és koncepciók

A játékfejlesztés lehetővé teszi 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 pygamet 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, 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 valakinek a játékfejlesztés terén, a Pygame és hasonló eszközök segítségével saját projekteket hozhat létre és járulhat hozzá a tudásunk bővítéséhez és a különböző területeken való alkalmazásukhoz. A játékok nagyon hasznos eszközök, amelyeket a gyerekek és felnőttek is élveznek. Rengeteg hasznos készséget lehet elsajátítani játszás során, ez a számítógépes játékokkal sincsen másképp. A kreatív és kihívást kedvelő embereknek a játékfejlesztés egy kiváló lehetőség arra, hogy kipróbálják magukat és megmutassák kreativitásukat. A játékfejlesztés során a fejlesztők számos készséget elsajátíthatnak, például a programozást, a dizájnt és a projektmenedzsmentet. A játékfejlesztés egyre népszerűbb a modern társadalomban, és egyre több ember kezd el játszani, ezért a fejlesztőkre is igen nagy az igény. Ez a fajta munka rengeteg lehetőséget kínál a kreativitás kibontakoztatására és a tanulás és fejlődés terén. A jövőben is szükség lesz a játékfejlesztésben jártas emberekre, hiszen ez örökké velünk fog maradni valamilyen formában.

2.5. Pygame és Ren'Py Összehasonlítása

A Pygame és a Ren'Py két olyan kiváló szoftverkönyvtár, amelyeket a játékfejlesztők és vizuális novellák készítői használnak világszerte. Bár első pillantásra talán nincse-

nek 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 lehetőséget teremtünk arra, hogy megtudjuk, melyik könyvtár a legalkalmasabb az adott célkitűzések és projektek szempontjából, 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

Általános célú keretrendszerként szolgál, ami azt jelenti, hogy szinte bármilyen típusú játék készítéséhez használható. Legyen szó akció-játékról, platformjátékról, vagy akár puzzle-játékról, a Pygame alkalmas erre. 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.

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 a visual novelek 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. Egyéb szempontok

A Pygame nyílt forráskódú és ingyenesen elérhető, valamint több platformon futtatható, beleértve a Linuxot, Windowst és MacOS-t is. Rugalmas és testreszabható, ami lehetővé teszi a fejlesztők számára a saját játékmotorjuk létrehozását. A Ren'Py mellett elérhető kereskedelmi licenc is, amely további támogatást és lehetőségeket kínál. Főként Windowson és macOS-en futtatható, és kifejezetten a visual novelek készítésére specializált, széleskörű szövegkezelési funkciókat nyújt.

2.5.7. Összegzés

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 visual novelek készítésére specializált keretrendszer.

A két keretrendszer kiválasztásakor a következő szempontokat érdemes figyelembe venni:

Játékstílus: Milyen típusú játékot szeretne készíteni? Tapasztalat: Mennyi programozási tapasztalattal rendelkezik? Költség: Mennyi pénzt szán a keretrendszerre? A megfelelő keretrendszer kiválasztása segíthet abban, hogy gyorsabban és könnyebben készítsen professzionális minőségű játékokat.

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 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

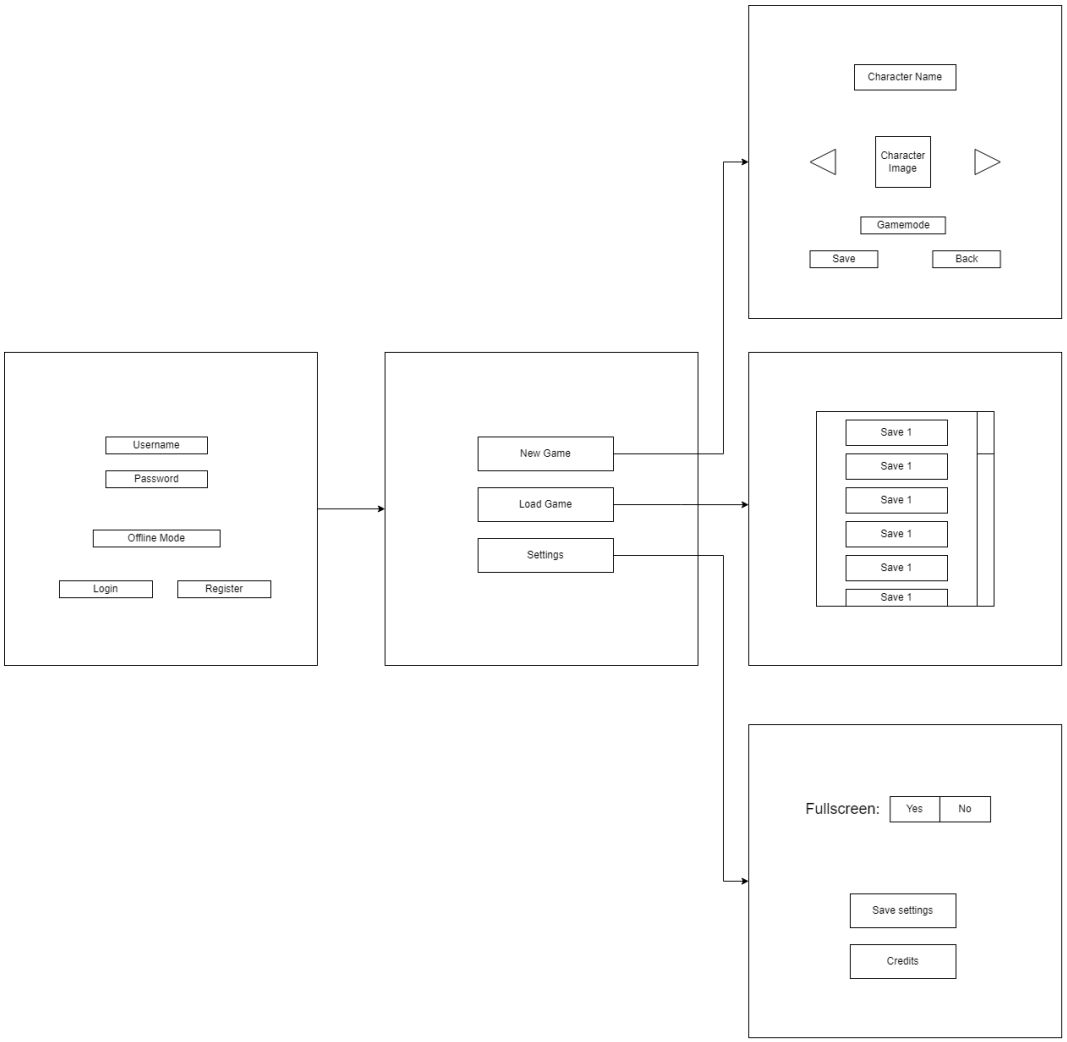
A játékok szempontjából a grafika kulcsfontosságú elem. Egy gondosan kidolgozott vizuális megjelenés mélyebben elvonja a játékosokat a játék világába, ami növeli a játék élvezetét.

Ezért fontos számomra, hogy saját magam készítsem el a rajzokat, ezzel is egyedivé téve a játékot, illetve a fejlesztés során egyszerűbben áthidalhatok majd a grafikai problémákat, 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

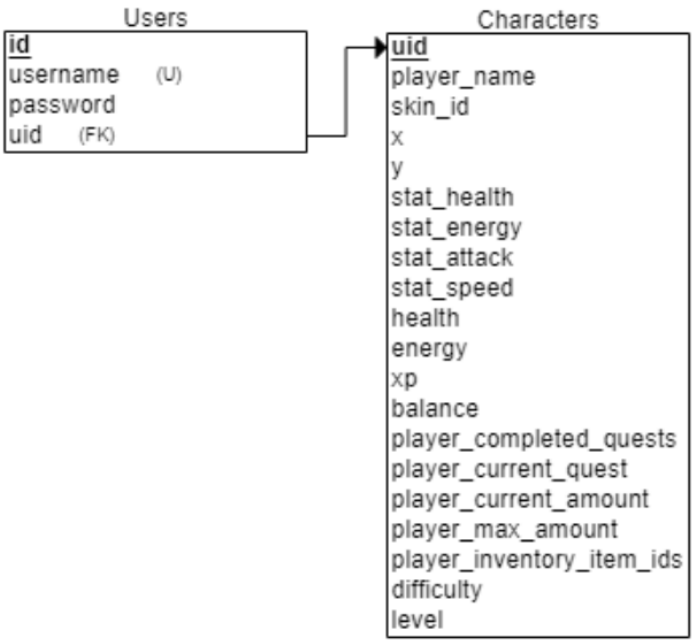
játékosok könnyen eligazodjanak benne, és ne kelljen sok időt tölteniük azzal, hogy megtalálják a keresett funkciót.



3.1. ábra: Menürendszer terve

3.3.1. Felhasználói fiók

Felhasználói fiók létrehozása azért szükséges a játékomban, mivel szeretnék létrehozni egy ranglista rendszert, ahol a játékosok tudnak egymással versengeni, ki hányszor vitte végig a játékot egyre nehezebb fokozaton. Ehhez a rendszerhez elengedhetetlen, hogy a játékosokat megtudjam különböztetni, illetve, hogy egy adatbázisban eltudjam tárolni a játékosok adatait.



3.2. ábra: Relációs modell

3.3.2. Főmenü funkciói

A bejelentkezést követően a játékosoknak három menüpont fog elérhetővé válni. Az egyik kiemelkedően fontos lehetőség a beállítások menüpont lesz, mivel általában itt találhatók a kulcsfontosságú funkciók módosítására szolgáló lehetőségek. Az én esetemben itt lesz elérhető a teljesképernyős mód beállítása. A beállítások menüpont alatt továbbá megtalálható lesz egy "Credits" opció is, ami tartalmazni fogja a készítő nevét és az elkészítés évét. A másik két menüpontnak pedig a játék indításával kell kapcsolatosnak lennie, mivel a játékosoknak lehetőségük lesz új karakter létrehozására, illetve meglévő mentésük betöltésére.

3.3.3. Mentés

3.3.4. Játékon belüli menürendszer

A legtöbb játékban kiemelkedő jelentőségű a játékon belüli pillanatfelvétel szolgáltatás, valamint egy belső menürendszer. Lényeges, hogy ezek könnyen elérhetőek legyenek, miközben nem zavarják a játékélményt. A játékosoknak lehetőségük lesz a játék közben is hozzáférni egy beállítások menühöz, amelyben a hangerőszintet is be tudják majd 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.

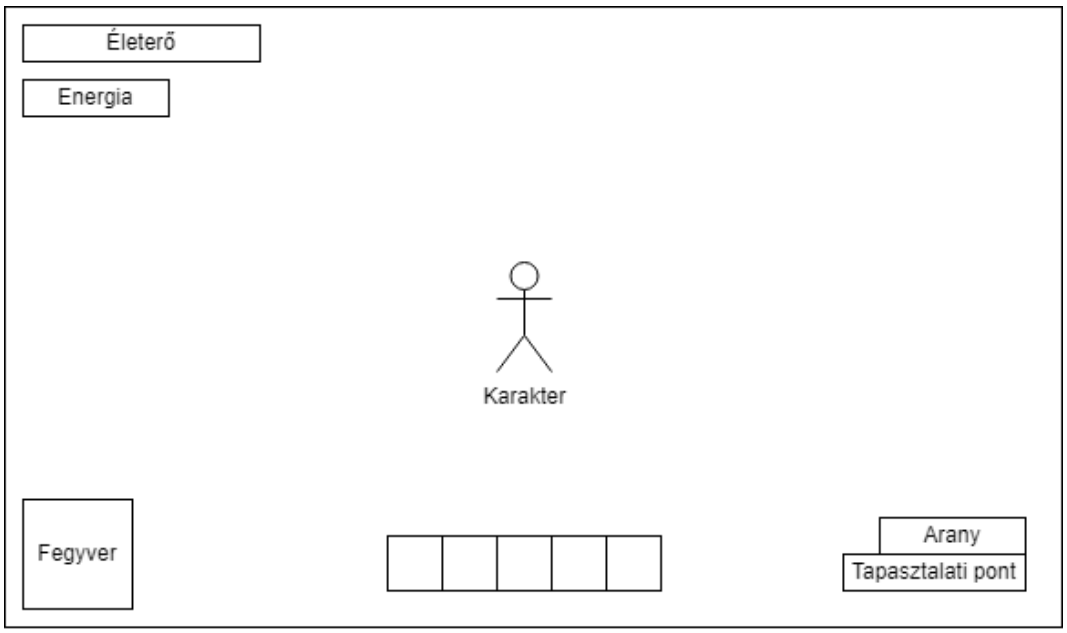
3.4. Felhasználói felület és a játékos cselekvési lehetőségei

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

3.4.1. Felhasználói felület

A felhasználói felület megtervezése során, igyekeztem 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 HUD (Head Up Display) a játék során folyamatosan látható lesz, így a játékosoknak nem kell külön menübe navigálniuk, hogy megtalálják a szükséges információkat. A HUD a játékos karakterének életerőjét, energia szintjét, a játékos aranyát, a játékos szintjét, a játékos tapasztalati pontjait, a játékos fegyverét, a játékos által használt italokat fogja megjeleníteni.



3.3. ábra: Felhasználói felület terve

3.4.2. Életerő és Energia

A játékos karakterének életerő (HP) mutatja, hogy mennyire életben van. Az életerő csökken, ha a karakter sérüléseket szenved, és elérheti a nullát, ami halált eredményez. Az energia pedig a karakter futáshoz és cselekvésekhez szükséges energiaforrás. A megfelelő kezelése és felhasználása kulcsfontosságú a túléléshez és a harc hatékonyságához.

3.4.3. Fegyver Választás

A játékosnak lehetősége van különböző fegyverek között választani, amelyek különböző képességekkel és támadási stílusokkal rendelkeznek. Ez lehetővé teszi számára, hogy testre szabja a harci stratégiáját és alkalmazkodjon az adott helyzethez.

3.4.4. Harcolás Szörnyekkel

A játék során a játékosnak számos szörnnyel kell megküzdenie. A harcok izgalmasak és változatosak, és a játékosnak ki kell használnia a karaktere készségeit 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 mélyüljenek 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. Italok Vásárlása és Használata

A játékos aranyat szerezhet a játék során, amit italokra is fordíthat. Az italok különböző hatásokkal rendelkeznek, például életerő visszaállítás vagy energia szint növelés. Az italok stratégiai használata a túlélés és a harcok kulcsa lehet.

3.4.7. Statisztikák Növelése XP-ből

A játékos karaktere XP-t 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 és a játékmenet testreszabását.

Ezen cselekvési lehetőségek összekapcsolva teremtenek egy gazdag és dinamikus játékélményt az Action RPG játékban, ahol a játékos döntései és cselekedetei hatással vannak a karakter fejlődésére és a játék világára.

4. fejezet

Megvalósítás

4.1. Grafikák elkészítése

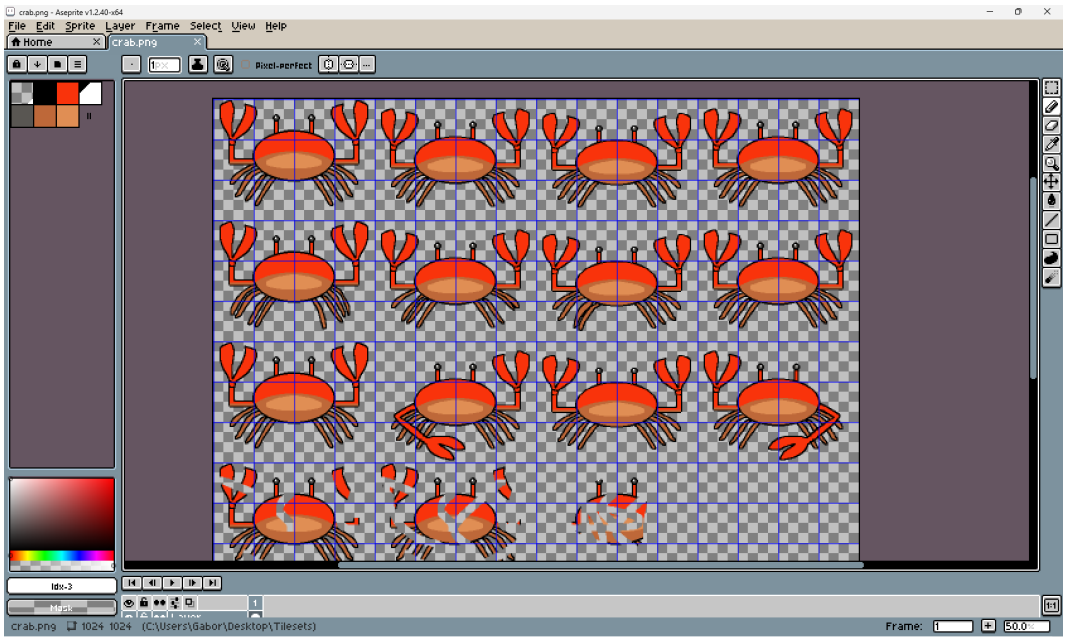
4.1.1. Grafika megtervezése

Inspiráció

Azok előtt, hogy 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.[5]

4.1.2. Grafikus szerkesztő

Fontos kiemelni, hogy egy "pixelart" rajz stílusról beszélünk az esetemnben, ezért fontos volt olyan rajzprogramot választanom amely minden igényemet kielégíti és könnyen kezelhető tapasztalatlan grafikusok számára is. A kiválaszott program az Aseprite volt, ebbe készítettem minden játékban megtalálható grafikát. [6]



4.1. ábra: Aseprite

4.1.3. 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 ki. 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 csempék harmonikusan illeszkedjenek egymáshoz, ezáltal kellemes és összefüggő látványt teremtve.

Entitások

Karakterek, és szörnyek megtervezése annyiban tért el, hogy ott figyelni kellett az animációra is, hiszen élethű mozgást szerettem volna elérni. 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 [7] 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 JSON 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.

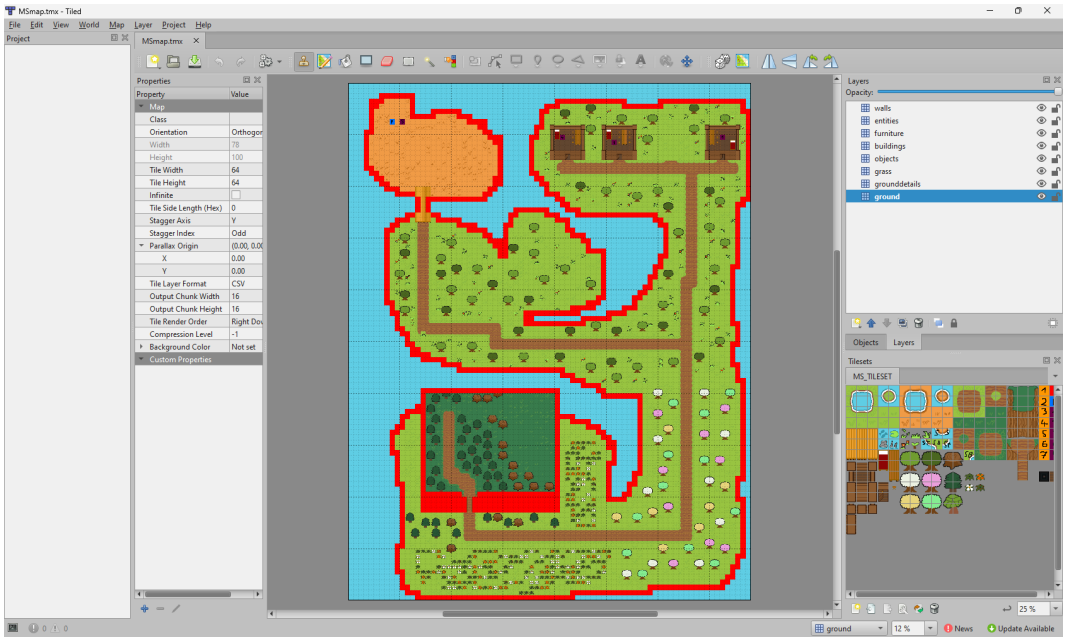
4.2.2. Tervezés

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 fogok használni.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.

Lettek még rétegek 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.



4.2. ábra: Tiled

4.3. Definiált osztályok bemutatása

A következő szekcióban a definiált osztályok kerülnek bemutatásra.

4.3.1. Settings

Ennek az osztálynak a 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 és menteni egy JSON fájlba.

Az osztályban található statikus osztályváltozók az alapértelmezett beállításokat tárolják, és a program különböző részeiben felhasználhatók. A statikus osztályváltozók előnye, hogy nem szükséges létrehozni egy példányt az osztályból, így könnyen hozzáférhetők.

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 (UI) színeit és stílusait, amelyek a játék megjelenítéséhez használatosak. Ezek a színek és stílusok segítenek a felhasználói felület személyre szabásában.

4.3.2. Game osztály

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 és élvezetes legyen, és a kódban 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)
```

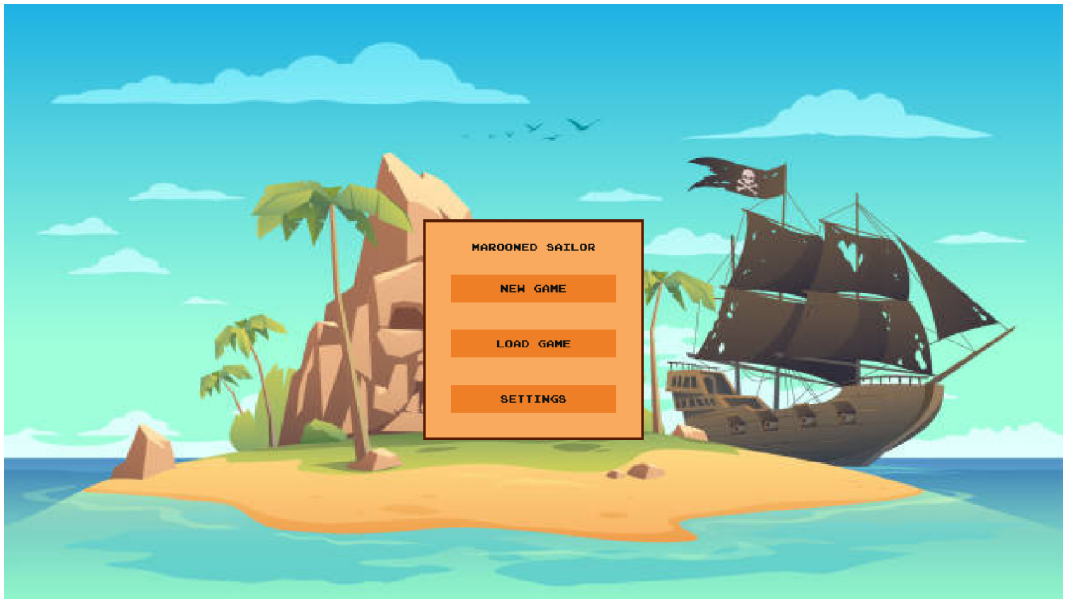
4.3.3. UserAuth

Szükségem volt, hogy adatokat adatbázisba mentsek és játékmenet közben frissítsem, elérjem azokat, ezért szükséges volt implementálni egy felhasználói fiók kezelő osztályt, amely FastAPI segítségével kommunikál a MYSQL adatbázissal. 4 fő metódussal rendelkezik, ezek a következők: felhasználónév, jelszó validálás, a bejelentkezés és regisztráció kezelése.

4.3.4. MainMenu

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

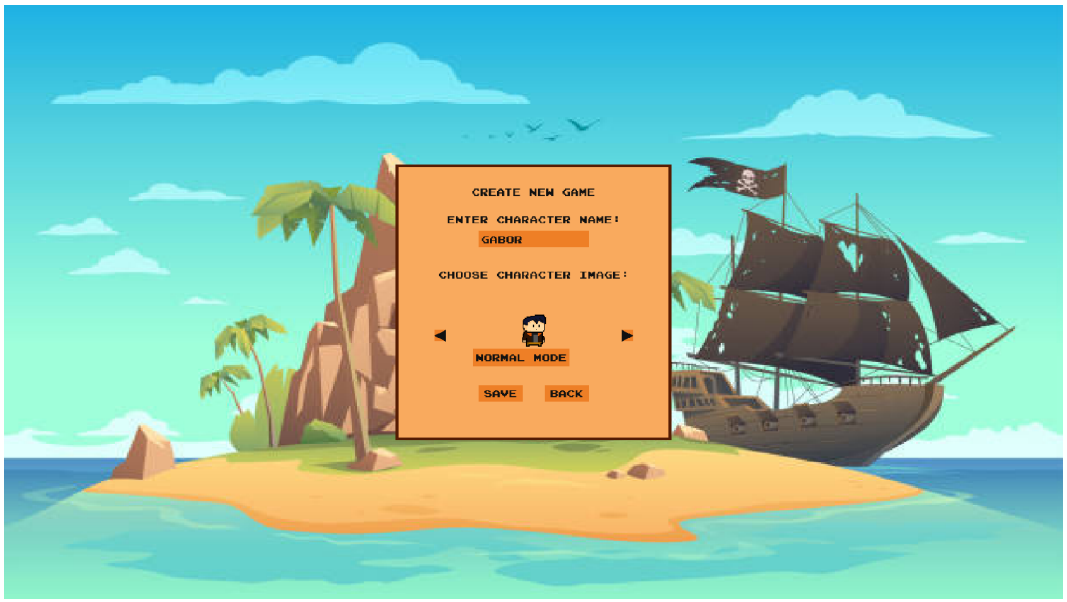
főmenüben három menüpont található meg, az új játék létrehozása, meglévő mentés betöltése, illetve a beállítások. (lásd. 4.3. ábra.)



4.3. ábra: Főmenü

4.3.5. NewGameMenu

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 elfogy a játékos életereje, akkor a játék folytatódik tovább, újraéled egy bizonyos helyen. Viszont, ha a "challange" módot választja akkor három funkcióval bővül a játékmenet. Az első ilyen funkció, hogy ha elfogy a játékos életereje akkor véglegesen meghal, 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 a legügyesebb játékosok hányszor tudták a "challange" mód használatával végigvinni az összes küldetést.



4.4. ábra: Új játék létrehozása

4.3.6. LoadMenu

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.

4.3.7. GameHandler

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 pillanatálj funkcióval együtt műkdöik, a ranglista, karakter fejlesztési menü, és még az irányítások felület is ahol a felhasználó megtudja nézni, melyik gomb mire való.

4.3.8. LevelHandler

Ez a másik nagyon lényeges osztály, ez a GameHandler-ben kerül példányosításra. Feladata az 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.

4.3.9. Entity

Ez az osztály egy szülő osztály, ami azt jelenti, hogy ebből származnak le más osztályok és felhasználják minden tulajdonságát illetve metódusait it. Három fontos metódussal rendelkezik az egyik a mozgás kezelése, 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 irányű hitbox-ához igazítjuk, így megakadályozva az áthaladást. (Lásd. 4.2 programkód)

Programkód 4.2. Ütközés kezelés

```
    def collision(self, direction):
    if direction == 'horizontal':
    for sprite in self.obstacle_sprites:
    if sprite.hitbox.colliderect(self.hitbox):
    if self.direction.x > 0:
    # moving right
    self.hitbox.right = sprite.hitbox.left
    if self.direction.x < 0:
                        # moving left
                        self.hitbox.left = sprite.hitbox.right

                        if direction == 'vertical':
                        for sprite in self.obstacle_sprites:
                        if sprite.hitbox.colliderect(self.hitbox):
                    if self.direction.y > 0:
                        # moving down
                        self.hitbox.bottom = sprite.hitbox.top
                        if self.direction.y < 0:
                        # moving up
                        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, hiszen 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.

4.3.10. Player

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, miközben sprintel. Az energia idővel regenerálódik, ami lehetővé teszi a karakter számára, hogy továbbra is használja a sprintet. A játékos karakter képes támadni is, és az attack metódus meghívásával hozza létre a támadást. A támadásokat az osztály figyeli, és számítja a támadások közötti hűlési időt. A karakter fegyvere is dinamikusan változik, és a játékos lehetőséget kap a fegyver cseréjére a játék során. A karakternek van egy egészségi és energia szintje, amelyek határozzák meg túlélését a játékban. Amennyiben az egészsége nullára csökken, a karakter meghal, de van lehetőség a visszatérésre vagy újrakezdésre a játékban, attól függően, hogy a játék nehézségi szintjétől függően. A karakter számos egyéb tulajdonsággal és képességgel rendelkezik, például tapasztalati pontokkal, pénzügyi egyenleggel 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 inventáriummal. 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.

4.3.11. World, Dungeon

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, hiszen 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. Hiszen 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 elválaszott 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, 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, hiszen 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) 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 4.3. csvtodict

```
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

Most, hogy már grafika és a pálya rétegek elkészültek amik ki lettek exportálva, ideje ezeket az adatokat egyesíteni és megjeleníteni a játékban. Ezt 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, előállítom az egyes képekhez vezető útvonalat, és ezután a pygame.image.load függvényével betöltöm a képeket. Ezt követően a képeket egy listába helyezem, 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 kiválasztom a megfelelő képet, és a megfelelő koordinátákat amiket segítségével példányosítom a Tile (csempe) osztályt később ennek segítségével végzem el a kirajzolást. Mivel a szótárak között szerepel az entitásokra vonatkozó réteg is, ezért azokat is az előbb említett iterációban kezelem, és példányosítom 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.

4.3.12. Enemy

Az ellenségek a játékban a küldetések után talán a legfontosabb mechanikái a játknak, hiszen 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 megállíptsa 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.

4.3.13. NPC

Nem játékos által vezértelt karakterek azaz NPC-k egy kalandjáték során elengedhetetlen részei a történetnek és a játékmenetnek. Ahogy az összes többi szörny, játkos osztály is az Entity osztályból származik le, hiszen adott esetben szükség lehet mozgatásra és ütközés érzékelésre. Azonban a legtöbb esetben a játékos karaktere nem tudja megölni ezeket az entitásokat, hiszen a történet során szükség van rájuk. Ezért a játékos karaktere nem tud ütközni velük, és a szörnyekkel ellentétben nem tudnak támadni.

4.3.14. Questgiver

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

A Questgiver NPC típúst 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 létrehozz 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 típúsú 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, hiszen 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.

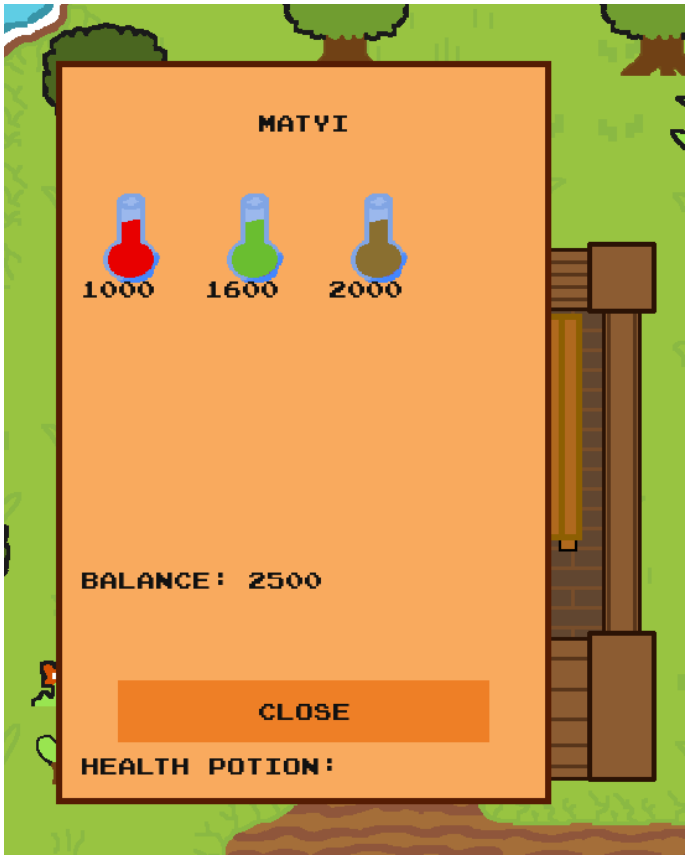
4.5. ábra: Dialógus

4.3.15. Merchant

Egy kalandjátékban az előrehaladás mellett fontos, hogy a nehezen megszerzett tárgyakkal, esetünkben az arannyal lehessen mit kezdeni, ezért egy Merchant NPC típús létrehozását láttam a legjobb ötletnek, hiszen rengeteg különböző módon fellehet őket használni.

Egyenlőre csak bájital árusításra szolgáló merchant van a játékban (élet visszatöltő, energiát növelő, sebzést növelő bájitalok árusítására szolgál). A továbbfejleszétsi lehetőségek bekezdésben jobban kifejtem mikre lehet még használni a merchant típúsú NPC-t. Ahogyan a Questgivernél is 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, típúsa, 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.



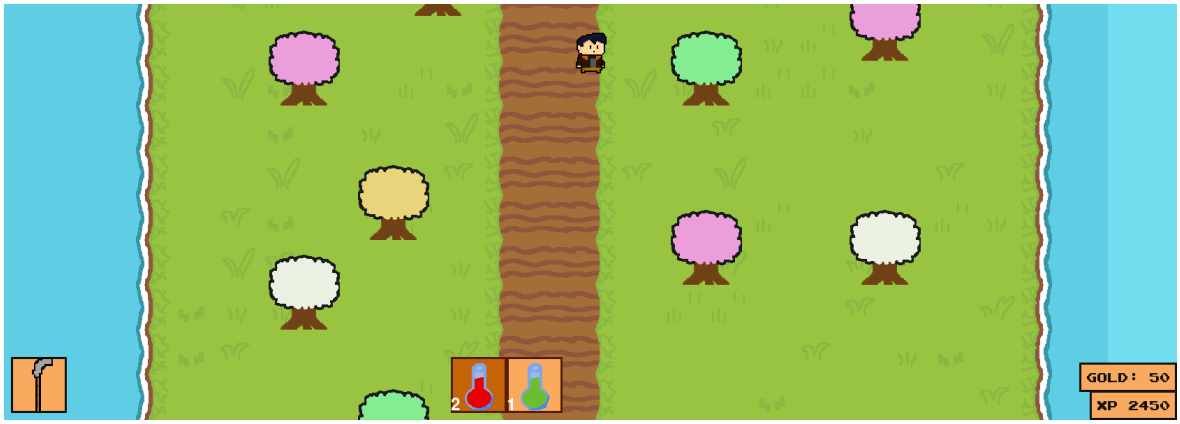
4.6. ábra: Merchant

4.4. inventory

Inventory azaz táska rendszer, fontos szerepet kap a játék folyamán hiszen a merchanttól megvásárolt tárgyak ebbe a hátizsákba kerülnek bele és a játékos ezeket felhasználni

amikor szükségét érzi.

Öt szabad férőhellyel rendelkezik a táska ezért jól meg kell válogatni mik 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, hiszen 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 döntöttünk, 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 tárgyból. (Lásd. 4.7 ábra)



4.7. ábra: Táksa rendszer

4.5. Loot

A zsákmányolás funkciót azért tartottam fontosnak, hiszen plusz motivációt nyújt a játékosnak legyőzni a szörnyeket, ha szerezhet 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 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 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, hiszen a Loots osztály ahogy a többi objektum is, a pygame.sprite.Sprite osztályból

származik le.

Tárgyak felvétele a tárgy pozíciója és a játékos karakterneket távolságát vizsgálva történik, ha elég közel megy a játékos a tárgyhoz akkor kitörli a kitörli a loots listából a felvett tárgyat és az értékeit hozzáadja a játékos karakterének a tulajdonságaiban tárolt értékekhez.

4.8. ábra: Zsákmányolás funkció

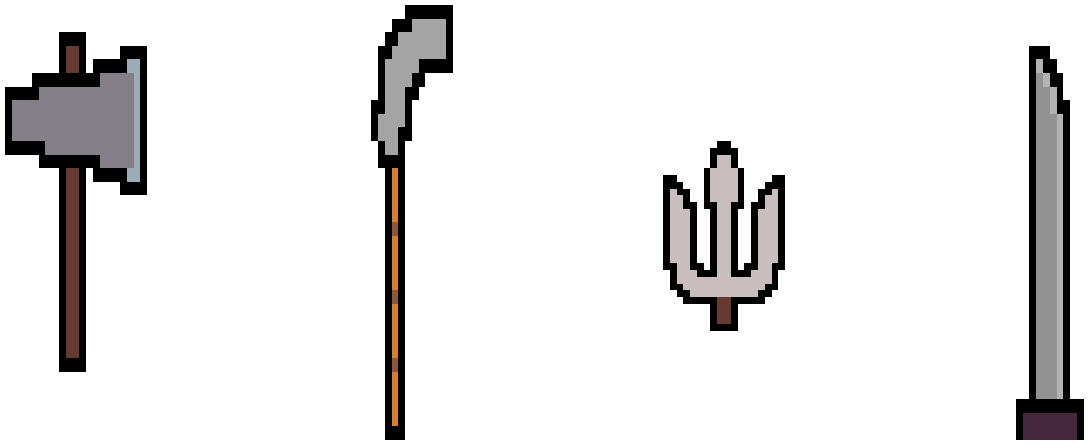
4.6. Weapon

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 frissités 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, hiszen 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.

Ahogyan a többi látható objektum ezek a fegyverek is a visible sprite csoport része, viszont rendelekzik 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.



4.9. ábra: A játékban megtalálható fegyverek

4.7. ingame_menu

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

A pillanatálj 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.

4.8. Save

Ahogy a korábbi fejezetekben már említettem a játékállapot mentését a 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-n keresztül kommunikálva egy mysql adatbázisban, ha online módot választott a felhasználó.

4.9. megvalisitando

tile particles tile projectile save Exportálás? game_api_client DOCKER BACKEND
weapon npc merchant questgiver user inventory loot
Legnyagobb kihívások amikkel szembesültem
továbbfejleszéts - fegyver szerzés opció / vásárlás, inventory növelés stb

5. fejezet

tutorial

Ez a fejezet mutatja be a megvalósítás lépéseit. Itt lehet az esetlegesen előforduló technikai nehézségeket említeni. Be lehet már mutatni a program elkészült részeit.

Meg lehet mutatni az elkészített programkód érdekesebb részeit. (Az érdekesebb részek bemutatására kellene szorítkozni. Többségében a szöveges leírásnak kellene benne lennie. Abból lehet kiindulni, hogy a forráskód a dolgozathoz elérhető, azt nem kell magába a dolgozatba bemásolni, elegendő csak behivatkozni.)

A dolgozatban szereplő forráskódrészletekhez külön vannak programnyelvenként stílusok. Python esetében például az 5.1. programkódban látható egy formázott kódrészlet.

Programkód 5.1. Python példa

```
    import sys

    if __name__ == '__main__':
    pass
```

A stílusfájlok a `styles` jegyzékben találhatók. A stílusok között szerepel még C++, Java és Rust stílusfájl. Ezek használatához a `dolgozat.tex` fájl elején `usepackage` paranccsal hozzá kell adni a stílust, majd a stílusfájl nevével megegyező környezetet lehet használni. További példaként C++ forráskód esetében ez így szerepel.

```
    #include <iostream>

class Sample : public Object
{
    // An empty class definition
    }
```

Stílusfájlokból elegendő csak annyit meghagyni, amennyire a dolgozatban szükség van. Más, C szintaktikájú nyelvekhez (mint például a JavaScript és C#) a Java vagy C++ stílusfájlok átszerkesztésére van szükség. (Elegendő lehet csak a fájlnevet átírni, és a fájlban a környezet nevét.)

Nyers adatok, parancssori kimenetek megjelenítéséhez a `verbatim` környezetet lehet használni.

```
    $ some commands with arguments
    1 2 3 4 5
    $ _
```

A kutatás jellegű témáknál ez a fejezet gyakorlatilag kimaradhat. Helyette inkább a fő vizsgálati módszerek, kutatási irányok kaphatnak külön-külön fejezeteket.

6. fejezet

Továbbfejlesztési lehetőségek

6.1. Táska rendszer

több item lesz kell több hely, és lehessen sortolni

6.2. Ellenségek

Több ellenség, egyedi képességekkel (most mind egyként működik)

A fejezetben be kell mutatni, hogy az elkészült alkalmazás hogyan használható. (Az, hogy hogyan kell, hogy működjön, és hogy hogy lett elkészítve, az előző fejezetekben már megtörtént.)

Jellemzően az alábbi dolgok kerülhetnek ide.

- Tesztfuttatások. Le lehet írni a futási időket, memória és tárigényt.
- Felhasználói kézikönyv jellegű leírás. Kifejezetten a végfelhasználó szempontjából lehet azt bemutatni, hogy mit hogy lehet majd használni.
- Kutatás kapcsán ide főként táblázatok, görbék és egyéb részletes összesítések kerülhetnek.

7. fejezet

Összefoglalás

Hasonló szerepe van, mint a bevezetésnek. Itt már múltidőben lehet beszélni. A szerző saját meglátása szerint kell összegezni és értékelni a dolgozat fontosabb eredményeit. Meg lehet benne említeni, hogy mi az ami jobban, mi az ami kevésbé jobban sikerült a tervezettnél. El lehet benne mondani, hogy milyen további tervek, fejlesztési lehetőségek vannak még a témával kapcsolatban.

Források

[1] Unity Technologies. *Unity Documentation*. 2023. URL: https://docs.unity3d.com/.

[2] Oracle Corporation. *Java Documentation*. 2023. URL: https://docs.oracle.com/en/java/.

[3] Microsoft Corporation. *C# Language Reference*. 2023. URL: https://docs.microsoft.com/en-us/dotnet/csharp/.

[4] The C++ Standards Committee. *C++ Standard Library Reference*. 2023. URL: https://en.cppreference.com/w/.

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

[6] Igara Studio S.A. *Aseprite*. 2014. URL: https://www.aseprite.org/.

[7] Thorsten Scheuermann. *Tiled*. 2023. 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).