Házi feladat specifikáció

Kalandjáték

A program leírása

A program egy egyszerű „kattingatós” kalandjáték, amiben a játékos különböző helyszíneken, különböző ellenfelekkel küzdhet meg bizonyos jutalmakért.

A kezelő felület 2 különböző gomb csoportból áll, név szerint **Utility** és **Interact** csoportok. Ezen felül áll még egy „**játék panelből**”, ahol a tényleges játék történik, illetve egy **Combat Log**-ból.

- A **játék panel** egy n*m-es mátrix, melynek egyes mezőiben ellenfelek, üres tér, tárgy vagy maga a játékos található. A táblázat hátteréül az adott helyszín képe szolgál. Egy adott mezőre kattintva kiválasztjuk azt.
- A **Utility** gombok segítségével érhetünk el olyan funkciókat, amik nem közvetlen a játéktérrel kapcsolatosak. Ilyen például a játékos felszerelésének módosítása, vagy a helyszínek közötti utazás.
- Az **Interact** gomb csoport segítségével tudunk kapcsolatba lépni a játék panelen kiválasztott mező tartalmával. Úgy, mint például odasétálni, megtámadni, felvenni. Az, hogy mit tudunk kezdeni az adott mezővel, az attól függ, hogy mi van benne (ellenség, tárgy, üres tér, stb…)
- A **Combat Log** egy chat szerű ablak, melyen a játékos a játék egyes eseményeit és adatait tudja nyomon követni. Ilyen esemény/adat lehet például a sebzések sorsolt mennyisége, az adott távolság a kiválasztott mezőtől, stb…

A program rendelkezik ezen felül egy **menü sorral**, ahol különböző, játéktól független tevékenységeket hajthat végre. Ilyen például a játék állásának mentése és betöltése.

A játék menete

A játék menete egy egyszerű loop-ból áll. A játékos rendelkezik felszereléssel, melyet alapból kap, viszont a játék során találhat jobb felszerelést. A felszerelés két részből áll: páncél és fegyverek.

A játék körökre épül, az ellenfelek előre megadott módon viselkednek a saját körükben. Minden entitás (játékos és ellenfél) mozoghat és támadhat egy körben.

Ha egy ellenség meghal, a játékos az ellenség által megszabott intervallumban véletlenszerűen kap jutalmat. A jutalom **Tapasztalat Pont** lehet.

A **Tapasztalat Pont** a játékos szintlépéséhez szükséges. A játékos szintje hozzáadódik a fegyverei sebzéséhez és a páncélja értékéhez, ezzel válva erősebbé.

A játéknak nincs előre meghatározott célja, viszont akkor ér véget, ha a játékos legyőzött minden ellenséget, minden helyszínen. Különböző helyszíneken különböző (életponttal és sebzéssel rendelkező) nehézségű ellenfelek találhatóak.

A program funkciói, use-case-ek

A játékos különféle tevékenységeket tud végezni a játék folyamán. Ezek a következők:

Mező kiválasztás
A játék során minden cselekvés előtt a játékosnak ki kell választania egy mezőt. Ezt olyan módon teheti meg, hogy rákattint az adott mezőre. Ilyenkor a játék kijelez pár információt az adott mezőről. Ilyen információ lehet a rajta álló ellenfél életereje és szintje, a mező távolsága vagy akár a rajta lévő tárgy típusa is. Az, hogy a játékos milyen akciókat hajthat végre, az a kiválasztott mezőtől függ. Azaz, milyen dolog tartózkodik rajta.

Harc
A harc akkor történik, amikor egy ellenség megtámadja a játékost vagy fordítva.

A játékos kiválasztja a mezőt, amin a megtámadni kívánt ellenfél tartózkodik és megnyomja az „Attack" gombot. Ekkor a játék prompt-olja, hogy melyik fegyverével kíván támadni és a választott fegyver effektív távolsága alapján megállapítja, hogy a támadás végrehajtható-e.

A harc során a két fél felszerelésének értékein múlik, hogy ki jön ki győztesként. A védekező a páncélja értékével véd, míg a támadó a fegyvere átütő erejével támad. A páncél értéke statikus, állandó érték, míg a fegyver átütő ereje megadott intervallumon belül random. (Pl.: 4-20-ig) Amikor egy fegyverrel támadunk, azt innentől „dobásnak" hívjuk, mivel a háttérben kocka dobálást szimulál a játék. A fegyverek sebzés és átütő erő intervallumait kockák reprezentálják. Például:

- *Longsword*: Hit: **+4**, Damage: **2d4+2**: A Hit (+4) érték az átütő erőhöz adódik hozzá. Az átütő erőre mindig egy 20 oldalú kockával dobunk és a dobott értékhez hozzáadjuk a fegyver Hit (átütő erő) értékét. Amennyiben túldobtuk az ellenfél páncélját, úgy dobunk a Damage (sebzés) kockákkal. A **2d4** azt jelenti, hogy **2 darab 4 oldalú kocka** dobott eredményét összegezzük. A **+2** az ehhez hozzáadott plusz értéket jelzi. A végleges eredmény a kiosztott sebzés.

A fegyverrel kiosztott sebzés a védekező fél életerejéből kerül levonásra. Felszerelés tulajdonságokról bővebben lejjebb.

Mozgás
Mozgásról akkor beszélünk, amikor a játékos egy üres mezőt választ ki. A kiválasztott mezőre ez után a „Move" gomb segítségével tud átlépni a játékos. Előtte viszont a játék összeméri a mező távolságát a játékos által egy körben megtehető távolsággal (sebesség) és csak akkor engedi végrehajtani a mozgást, ha a játékos aktuális „sebessége" nagyobb vagy egyenlő a mező távolságával. A bejárt mezők száma levonódik a játékos aktuális sebességéből. Addig mozoghat, ameddig még rendelkezik 0-nál több sebességgel. A játékos sebessége visszaállítódik az alap értékére a köre elején.

Tárgy felvétel
A tárgy felvétel akkor hajtható végre, ha egy mezőre egy tárgy lett helyezve. Ebben az esetben a játékos odamehet és egy mező távolságból felveheti a tárgyat, ami így a felszerelései közé adódik.

Utazás
A játékos ezen funkció segítségével tud közlekedni helyszínek között. Minden helyszínen meghatározott számú, erősségű és pozíciójú ellenfél van, ezentúl más-más háttér társul a különböző helyszínekhez. A legyőzött ellenségek mentésre kerülnek és az adott helyszínre való visszatéréskor nem élednek újra. Az

Utazás funkció a Utility gombok között található és feldob egy panelt, melyen az elérhető helyszínek közül választhatunk.

Felszerelés kezelés
A játékos nyomon követheti a felszerelését és le is cserélheti azokat a játékban találtakra az „Equipment” gomb segítségével (Utility gombok). A játék panelen előfordulhatnak tárgyak a földön, melyekre kattintva azok felvehetők. A különböző tárgyak különböző erősséggel rendelkeznek. A páncéloknak **mozgási bónusza** és **védelme** van, míg a fegyvereknek **átütő ereje**, **sebzése** és **effektív távolsága**.

- **Mozgási Bónusz:** A játékos által egy körben *bejárható mezők számát* növeli.
- **Védelem:** A játékos *védelmét* jelzi, az ellenséges fegyverek *átütő erejének* ezt kell „túldobni”. A védelemhez hozzáadódik még a használó szintje is, így fejlesztve az páncélt.
- **Átütő erő:** A fegyver azon értéke, ami a védelemre dobott kocka eredményéhez adódik. A kettő érték összegének kell nagyobbnak, vagy egyenlőnek lennie az védekező fél *védelmével*.
- **Sebzés:** A fegyver által meghatározott intervallum, ami a sebzés dobás minimum és maximum értékét hivatott jelezni. A dobott eredményhez hozzáadódik a használó szintje.
- **Effektív távolság:** Azt jelzi, hogy a fegyvert hány mezőnyire lehet használni. Például egy íj használható akár 5-6 mező távolról is, míg egy kard maximum 2.

Egyéb esetek:

Mentés
A játék állapota bármikor elmenthető és betölthető. Ez a játékos feladata, a fenti menü sorból tudja megtenni. A játék lehetőséget biztosít több mentés létrehozására, így a játékos szabadon hozhat létre mentéseket és töltheti be azokat, több mentési állapotot tarthat fent.

Ellenség legyőzése
Egy ellenséget akkor tekintünk legyőzöttnek, amikor az életereje eléri a 0-t. Egy ellenség legyőzése mindig valamennyi Tapasztalat Pont szerzésével jár. Ez a Tapasztalat Pont a játékos szintjének növeléséhez szükséges. Elegendő Tapasztalat Pont szerzése utána a játékos szintet lép.

Játékos halála
A játékos akkor hal meg, ha az életereje eléri a 0-t. A játékos halálakor a legutolsó mentés kerül betöltésre.

Megoldási ötletek:

Játék Panel: A játéktér két különböző mátrixból áll (későbbiekben „grid”) és egy háttérképből. Mind a háttérkép, mind a grid-ek n*m-es értéke fájlból olvasható. A két grid egymáson helyezkedik el és a felső egyforma gombokat, míg az alsó az ellenfelek, tárgyak és a játékos képeit, „figuráit” tartalmazza. Amikor egy adott helyszínt betölt a program, beolvassa egy fájlból a helyszínhez tartozó hátteret, a vízszintes és függőleges gomb mennyiséget és az ellenséges pozíciókat, azok típusait.

A két grid számontartja a benne elhelyezett komponensek pozícióit. A felső grid-en megnyomott gombok visszaadják a gridben elfoglalt pozíciójukat, ami alapján a program megkeresi az alsó gridben ugyanazon pozíción tartózkodó entitást. Az entitás típusától függően lehet azzal kapcsolatba lépni (támadni, felvenni, mozogni, stb).

Mentés/Betöltés és pálya elemek: A játékállás mentése a karakter adatok és az alaphelyzettől eltérő ellenfél adatok JSON fájlba történő mentésével hajtódik végre. Minden tárgy, ellenfél és helyszín egyedi azonosítóval rendelkezik, ezekre hivatkoznak egymás között.

Például egy helyszín adatait JSON formátumban egy fájlban tároljuk. Ez a helyszín referál a rajta található ellenségekre azok azonosítóival. Ezek az ellenfelek pedig a saját JSON fájljukban helyezkednek el. Ezzel elkerülhető, hogy az elemek hardcode-olva legyenek a programba, minden asset egy-egy fájlból olvasható.

Az újra felhasználhatóság miatt létezik egy ASSET_ID és egy INSTANCE_ID. Az ASSET_ID segítségével azonosítjuk az ellenségeket és egyebeket tároló fájlokat, az INSTANCE_ID-val pedig az azonos típusú dolgokat tudjuk többszörözni. Például van egy Ork nevű ellenségünk, aminek az ASSET_ID-ja *ork-type-01.* Viszont egy helyszínen szeretnénk több ilyen típusú Orkot is elhelyezni. Erre megoldásként a helyszínt leíró JSON-ben egyedi INSTANCE_ID-kkal soroljuk fel az Orkokat. Például:

- ork-type-01: {cave-ork-1, cave-ork-2, cave-ork-3}

Ezzel elérjük, hogy ne kelljen fájlt duplikálni, csak mert ugyan olyan ellenségből többet szeretnénk használni.

Azért van szükség további INSTANCE_ID-ra, mert például a játék állásának mentésekor ezekkel hivatkozunk azokra az ellenfelekre, akiket a játékos már legyőzött, ezek a mentés újra töltésekor már nem kerülnek elhelyezésre a grid-en.

A program indítása után specifikálni kell egy fájlt, amiben a különböző pályaelem fájlok elérési útjai és típusai találhatók.

A program felépítése: A program különböző absztrakciós szintekre van bontva feladatok szempontjából. Ilyen szint például a fájlkezeléssel foglalkozó osztályok szintje, vagy a UI komponenseket kezelő osztályok szintje.

Az kisebb feladatokat ellátó osztályok, mind a saját absztrakciós szintjükön helyezkednek el, nem foglalkozva a „külvilággal”. Például egy UI komponenseket kezelő osztálynak nem kell tudnia hozzáférni a játékos jelenlegi sebzés értékéhez.

Amennyiben mégis szükség lenne rá, akkor az úgy nevezett „kapcsoló osztályok” segítségével férhet hozzá közvetetten. Ezek az osztályok a „szintek között léteznek”, ugyanis foglalkozhatnak egyszerre 2 vagy több absztrakciós szint közötti adatmozgatással is.

A nagyobb, összetetteb program logikát megvalósító osztályok egy része a *singleton* módszert alkalmazza a könnyebb kezelhetőség érdekében.

A különböző asset-eket, mint az ellenségek, tárgyak és helyszínek úgy nevezett „storage” osztályok tárolják, melyek egyetlen célja ezen adatok nyilvántartása.