Donald - A Kalandjáték

Pap Ádám

Feladatkiírás:

Kalandjáték

Tervezzen objektummodellt kalandjáték objektumainak leírására! Legyen pálya, harcos, szörny, és legyenek tárgyak, amit a harcos fel tud szedni. Határozza meg az objektumok kapcsolatát és felelősségét!

Demonstrálja a működést külön modulként fordított tesztprogrammal! A játék állását nem kell grafikusan megjeleníteni, elegendő csak karakteresen, a legegyszerűbb formában! A megoldáshoz ne használjon STL tárolót!

Bevezető:

Felleges Ódák Tava. A csendes, bukolikus táj azt hiteti el az emberrel, hogy minden rendben van, azonban az útja során rájön mindenki egyre:
a vidéket veszélyek lakják.

Réges rég, a tájkép még más volt, fegyelem volt, becsületesség. Aztán a semmiből Tódor, a Tolvaj előbukkant és felzargatta a vidéket.
Hozta magával két csatlósát is, balkezét Günthert, a Góliátot és jobbkezét, Robit, a Rosszfiút.

Segélykiáltást hallottál messziről - gondoltad ki lehet az? Ki más, szegény Egon éppen sírva közölte veled, mit műveltek a házával a tolvajok. Arra kért meg, hogy bosszuld meg családját. Te, mint kalandor, egy ilyen küldetésnek nem állhatsz ellen. Harcra fel!

A játék környezete/menete:

A konzolon tudja majd a játékos irányítani az eseményeket.

A történet fájlból lesz beolvasva, mely a játékost kísérni fogja kalandja során, előkészítve a megfelelő környezetet egy kalandjátékhoz. A játékmenetet a játszma során el lehet menteni fájlba, és a játék elején be is lehet tölteni azt.

Az események a következőek: 3 főgonosszal kell majd megküzdeni, azonban a harcra fel kell készülni! Így minden harc előtt lesz lehetőség tárgyakat szerezni, amely megnövelheti a sebzését a kalandornak, illetve az életerejét. A felkészülés küldetésekből fog állni, ezek nem mások, mint Egon matek házijai!

Mindig lesz három feladvány egy harc előtt, ezek megoldásával tud a játékos tárgyakat venni. Ha a feladvány jó, a kalandor kap érmét, ha nem akkor tájékoztatva lesz arról. A harc előtt felugrik a bolt, amelyből lehet tárgyakat venni, (az érmék függvényében, ha valamire nincs elég érme akkor hibaüzenetet kap a játékos) majd kezdődhet a harc, mely folyamán a játékos támad, majd a szörny, addig amíg az egyik el nem pusztul.

Vezérlés:

A játék vezérlése igen egyszerű, a konzol segítségét veszi igénybe:

- automatikusan megjelennek a történetrészek, ekkor a kalandornak csak olvasnia kell, nem kell semmit interaktálnia a konzollal
- küldetések során a játékosnak be kell írnia a megfelelő eredményt a válaszmezőbe, ez egy egész számot fog elvárni eredményül
- a boltban felsoroltathatja a játékos a tárgyakat, amiket megvehet, ezek tulajdonságait ki tudja majd listázni, itt a bolt megnyitása „1”-es gombra fog megtörténni, ekkor felsorolódnak a tárgyak nevei, s sorszámai, ezután a játkos „x y” formátumban megadhatja melyik tárggyal mit szeretne csinálni
    - x – 0-9-ig számok, melyek a tárgyak sorszámait jelölik
    - y – 0 vagy 1 -> ha 0-t ír akkor a tárgy tulajdonságai kiíródnak, ha 1-et akkor pedig megveszi azt (természetesen, ha van rendelkezésre álló érméje)
- a bolt szekcióból vissza lehet menni „-1” beírásával, majd „2”-es gombra a játékos elindíthatja a harcot a szörnnyel, mely szintúgy interaktálás nélkül lejátszódik a konzolon, és végig követheti a kalandor a sebzését/sebződését
- a különböző szekciók során lehetősége lesz a játékosnak a játék mentésére, ezt egy 10-es szám begépelésével tudja megvalósítani

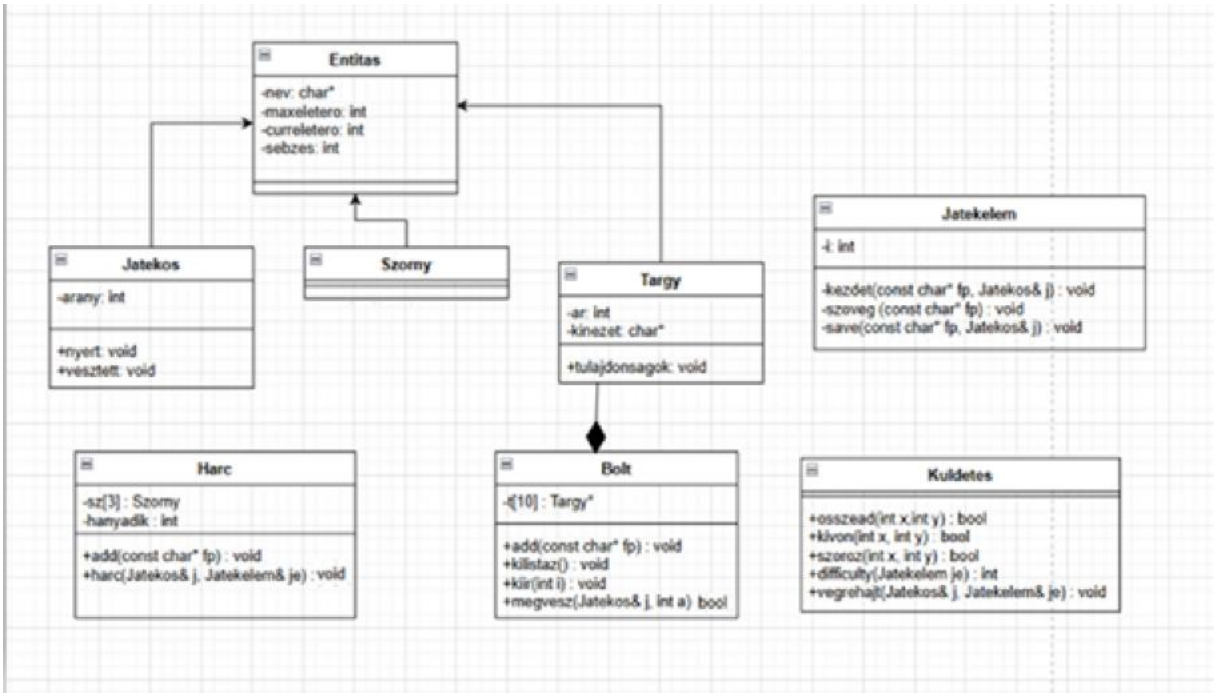
Ha a játékos rossz formátumban ad meg valamit, arról tudomást fog adni neki róla a program.

A játék vége:

A játéknak akkor lesz vége, ha az utolsó szörny is (Tódor) le lett győzve, avagy, ha a játékos életereje 1 alá csökken. Erről tájékoztatva lesz a játékos. (Vagy ha elmenti a játékot természetesen.)

Megvalósítás – Terv:

UML ábra az osztályokról



Az UML ábrán nem tüntettem fel az adott osztályokhoz tartozó konstruktorokat/destruktorokat, illetve a setter/getter függvényeket, ezeket természetesen implementáltam.

Magyarázat, fontosabb megvalósítási lépések:

Megvalósításhoz egy állapotgépet fogunk használni, melyben az i értéke fogja meghatározni, hogy mely állapotban vagyunk.

Az elején létrehozzuk a jatekelemet, kezdetben az i-nek 1-es értéket fogunk adni. Az i fog minket végig vezetni a játékon.

1, ha i % 4 == 1,2,3 akkor tudjuk, hogy küldetést kell meghívnunk

2, ha i % 4 == 0 akkor feljön az opció a bolthoz

3, majd ugyanebben az állapotban, elindítható a harc is

Megj: Egy enummal fogja eldönteni hogy melyik állapotban vagyunk, ez az enum Allapot.

1, A küldetés igen egyszerű, random szám alapján döntjük el a vegrehajt() függvény melyik matematikai példát kapja meg, és ismét sorsolunk két számot amellyel el kell végeznie az adott műveletet. Az „i” a nehézséget fogja állítani, ezt a difficulty függvénnyel határozzuk meg pontosabban.

2, A boltban tároljuk a Targyakat, melyeket ki tudunk listázni a kilistaz() függvénnyel, és ha a Jatekos-nak van elég aranya akkor meg is tudjuk venni azt. Ha csak meg akarja tekinteni, arra a kiir fgv. szolgál.

3, A harc során, ha i / 4 == 3, akkor a játéknak vége van, meghívhatjuk a nyert() függvényt. Ha a játékos meghal, akkor pedig a vesztett() függvény hívódik.

Érdemes megemlíteni, hogy az i++ megtörténik egy küldetés és harc után is, így mozdul előre a játék.

A Jatekelem osztály további függvényei a fájlba mentés, betöltés, és a történet kiírását teszik lehetővé.

Az Entitas osztályban, a játékosnak a pénze megnövekszik a helyes küldetések teljesítésével. A Targyaknal el van tárolva a kinézet, ez fogja megmutatni hogy néz ki az adott tárgy. A tulajdonsagok fgv-t meghívva ki tudjuk sorolni egy adott tárgy összes tulajdonságát. A tárgyaknál, habár zavaró lehet, de nincsen „életerejük”, ezt bónusz életerőként kell értelmezni, amely a játékoshoz fog hozzáadódni (ugyanez sebzéssel). A szörnyek a Harc objektumba vannak tárolva egy listában, ez azért előnyös mert a harcok során ők fognak majd kelleni. (itt az int hanyadik fogja megmondani melyikkel harcolunk épp).

Tesztekhez megjegyzések:

A tesztesetekben csak a konstruktorokat és az alap setter/getter függvényeket teszteltem. A további tesztekhez úgy érzem, hogy a filekezelést kell elvégeznem, ami a következő, és kb utolsó lépése a kész-nek. Amin még gondolkoztam az a String implementálása, azt úgy éreztem, hogy az expect_str-t így tudnám megoldani (pl. nevek jól álltak-e be stb.). Még, amit majd meg szeretnék kérdezni laboron az az hogy pl. a Kuldetes class működését hogyan tudom tesztelni? Itt ugye user input kell, s így nem látom, hogy hogyan lehetne.