Contents

Pontosított specifikáció

Feladatkiírás

Készíts egyszerű toronyvédő (tower defense) típusú játékot az SDL könyvtár segítségével! A játék lényege, hogy ellenfelek jönnek egy előre látható útvonalon, és a játékos által stratégikusan lehelyezett tornyoknak kell megakadályoznia a célba jutásukat. Tornyokat az ellenfelek megölésével szerzett pénzből lehet venni. A játéknak akkor van vége, amikor egy meghatározott számú ellenfél eljut a célba. A pályát a játék fájlból olvassa be; lehessen a legmagasabb elért szintet (pontszám) névvel elmenteni, és a mentetteket visszatekinteni.

Főmenü

A játék rendelkezik egy főmenüvel, ahonnan az Új játék gombbal el lehet indítani a játékot. A főmenüből lehet az irányításnak utánaolvasni és a dicsőséglistát megtekinteni, ezek a főmenü jobb oldalán lesznek elérhetők. Weblinkek találhatók továbbá a program github repójára, illetve a használt képkészletre.

A játéknézetben van egy koponya, amivel rákérdezés nélkül visszatérünk a főmenübe, mentés nélkül.

Játék menete

Pálya

Az ellenségek előre meghatározott, egyértelmű utat követnek (1 helyen jönnek be és 1 helyen mennek ki), a pálya többi részére pedig tornyok lerakhatók. A játék a pályát egy fájlból olvassa be, ahol 1-1 karakter 1-1 csempének (tile) felel meg. A pálya mérete 20×15, fix. Az ellenségek által bejárt út keresztezheti önmagát, de eleje és vége csak a pálya tetején lehet.

A lehetséges pályaelemek:

- *szóköz*: semmi
- `.`: padló, torony lerakható
- `,`: padló, torony nem rakható le
- `k`: ellenségek kezdőpontja, csak egy lehet belőle, felül kell lennie
- `u`: út, `k`-tól kell kezdődnie, egyértelműnek kell lennie, felül kell végződnie, egyéb kritériumok
- `o`,`O`,`f`,`s`,`S`,`t`,`T`: bal, jobb, alsó-felső falak, sarokelemek

A játékkal 1 db pálya csomagolva lesz.

Ellenségfajták

- Ellenség -> mi lesz belőle, ha lelövik
- Lelkecske -> meghal
- Kis Ogre -> Lelkecske
- Kis Démon -> Lelkecske
- Nagy Ogre -> Kis Démon
- Nagy Démon -> Kis Ogre

Habár kevéssé intuitív ez a felépítés, szeretném, hogy legalább egy picit kelljen taktikázni, hogy hova milyen tornyot tesz le az ember.

Toronyfajták

- Íjász, nyilat dob, és csak a Lelkecskét tudja lelőni
- Lovag, dárdát hajít, és a Lelkecskét és az Ogrékat tudja lelőni
- Varázsló, vizesüveget hajít, és csak a Démonokat tudja lelőni

A tornyok nem fejleszthetők.

A toronykiválasztó felületnél, ami a képernyő alján van, egyértelműen van jelezve, hogy melyik torony mit tud lelőni.

Látótávolság

A tornyok csak a látótávolságukon belüli ellenfeleket célozzák, a látótávolságot lerakáskor lehet látni.

Balanszolás céljából nem lehet úgy tornyot letenni, hogy a tornyok lássák egymást.

Pénz és élet

Minden ellenség lelövésekor kap a játékos 1 érmét (összességében: Lelkecske: 1, Nagy Ogre: 3, stb), ezekből tud tornyokat venni. Ha az ellenség eljut a célba, akkor annyi életet vesz le, amennyit még bele kellett volna lőni (Lelkecske: 1, Nagy Ogre: 3, stb).

Hullámok

A játék kör alapú, a hullámokban idézendő ellenségeket fájlból olvassa be. 1-1 karakter 1-1 ellenséghez tartozik: `1`,`o`,`d`,`0`,`D`,*szóköz* a lehetséges értékek (szóköz: kis idő kihagyás). A hullámok (fájlban sorok) száma és a hullámokban idézendő ellenségek száma is tetszőlegesen nagy lehet. Ezen felül a játék végteleníteni tudja a hullámokat az előző hullámok mindenféle kombinációjával, determinisztikusan.

A játékkal egy adag hullám csomagolva lesz.

A játék vége

Ha a játékos élete elfogy, elveszti a játékot. Ekkor lehetősége van egy nevet megadni, amivel fel tud kerülni a dicsőséglistára, ha az eredménye indokolja. A pontozás az alapján megy, hogy a játékos melyik hullámban vérzett el.

Játékállás menthetősége

A játékállás nem menthető.

Irányítás

Bal egérgombbal a kiválasztó felületre, vagy az 1, 2, 3 gombokkal választható ki a leteendő torony, és bal egérgombbal tehető le. Escape gombbal, vagy még egy bal gombbal megszüntethető a kiválasztás. A név megadása Enterrel fejeződik be, és a nézetek között szintén bal gombbal lehet váltani a megfelelő gombokra kattintva.

Dicsőséglista

A max. 6 elemű dicsőséglistát a játék fájlba menti, a következő formátumban, egy-egy sor egy-egy elért eredmény:

`elért_legmagasabb_hullám név akár szóközzel`

A fájlban pontszám szerint csökkenő sorrendben vannak az eredmények. A név hossza erősen limitált, 20 karakterből állhat csak (hogy kiférjen oda, ahol meg lesz jelenítve). Szóköz, a-z, A-Z, 0-9, ékezetes karakterek és egyes írásjelek is használhatóak a névben.

Hibás fájlok

A játék hibás fájlok lekezelésére nem vállalkozik, de igyekszik az első megtalált hibánál értelmes, a hibára rámutató hibaüzenetet kiírni. A program ekkor bezáródik, vagy helyes működés ezután (általában) nem várható el.

Kinézet

Textúrák

A játék 0x72 Dungeon Tileset II-jének a textúráit használja a kinézethez.

Játékablak

Az ablak tetszőlegesen átméretezhető, de a kép csak 320×240 képpont többszörösére fog felnagyulni, a maradék hely feketeséggel lesz kitöltve.

Apró módosítások

Egyes helyeken egy picit átfogalmaztam, pontosítottam a specifikációt, de lényegi módosítás a beküldötthöz képest nem történt.

Programozói dokumentáció

Nyelvhasználat

A program változó- és függvénynevei a tárgy példaprogramjaihoz hasonlóan magyar nyelvűek, azaz `*p` mint pointer helyett `*m` mint mutató, `Vector2` helyett `Vektor2`, `renderer` helyett `megjelenito`, stb. lesz olvasható. Az `am` az állománymutató (állomány=fájl) rövidítése.

Fordítás

A program fordításához szükség van az SDL2 könyvtárra és alkönyvtáraira. Ezen és a standard könyvtáron kívül más könyvtárra nincs szükség.

Linux

Ubuntun a következőképp lehet az SDL2-t telepíteni:

```
sudo apt install libsdl2-dev libsdl2-gfx-dev libsdl2-image-dev libsdl2-mixer-dev
```

A fordításhoz a `fordit_es_futtat.sh`-t, vagy valami ahhoz hasonló parancsot kell futtatni. Programhibamentesítéshez a `debug_fordit_es_futtat.sh`-t lehet használni.

Keresztfordítás Windowsra

Ubuntun a MinGW-t a következőképp lehet telepíteni (lehet, hogy valami más is kell még hozzá ezen kívül):

```
sudo apt install gcc-mingw-w64
```

Ez után az SDL-t az InfoC-ról a legegyszerűbb letölteni: az SDL telepítős oldalról, vagy közvetlenül.

Ebből másoljuk ki a `MinGW` mappát a `forraskod` és a `tartozekok` mellé, használjunk a `windows_fordit.sh`-hoz hasonló parancsot és reménykedjünk. Ha létrejött a futtatható fájl, akkor még a DLL-eket a `MinGW/bin` mappából át kell másolni a futtatható fájl mellé, hogy megtalálja őket.

Használt szavak

A programkódban vannak bizonyos gyakran használt szavak, amelyek jelentése elsőre nem feltétlenül egyértelmű:

- csp: csempe pozíció rövidítése, az adott csempének (tile) a pozíciója (koordinátái), egész szám, például a 2. oszlop 4. sorában: `v2e(1, 3)`, Vektor2e típusú
- kép: egy adott textúrára utal a `kepkeszlet.png` fájlból, egy hosszú felsorolt típus a `keprajzolo` modulból
- `jatek_ff`: a játék felhasználói felülete (ui), nem tartozik hozzá a bemenet kezelése
- dísz: olyan torony, lövedék vagy ellenség, ami nem mozog, csak (a képernyő alján) a felhasználó tájékoztatását szolgálják, díszek
- fs: függőleges szinkronizáció, vsync
- kps: kép per másodperc, fps
- `*_frissites(double delta)`: minden képkockánál meghívódó függvények, általában állapotok frissítéséért (pl. lövedék esik) és kirajzolásért felelnek
- delta: az előző frissítés óta eltelt idő másodpercben (pl.: 0.0167 s), egyszeres sebességnél a ténylegesen eltelt idő, de a játék így állandó képkocka per másodperc esetén is tetszőlegesen gyorsítható, illetve bármilyen kps-sel játszható
- `*_inicializacio()`, `*_destrukcio()`: (általában) paraméter és visszatérési érték nélküli függvények, amik a modul belső dolgainak létrehozásáért és eltüntetéséért felelnek, vagy a program elején és végén, vagy nézetváltáskor kell meghívni
- `szerez_*()`: egy statikus globális változó értékét adja vissza külsősöknek (getter)

Modulok, adatszerkezeteik és legfontosabb függvényeik

A függvények pontosabb leírását a fejlécfájlok, illetve statikus függvények esetén a forrásfájlok tartalmazzák.

foprogram

Meghívja bizonyos modulok inicializációját, benne található a főciklus, majd a modulok destrukcióját is meghívja. Az `idozites` segítségével kezeli az időzítést, illetve feldolgozza a parancssori argumentumokat.

idozites

A delta kiszámításáért, a játék gyorsításáért és a kps számolásáért felel.

nezetek

A frissítést és a bemenetet kapja a főprogramtól, és továbbküldi az aktuális nézetnek. Nézetek váltásakor meghívja az előző nézet destrukcióját és a következő inicializációját.

fomenu_nezet, jatek_nezet, jatekvege_nezet, iranyitas_nezet, dicsoseglista_nezet

Megkapják az inicializációt, frissítést, bemenetet és destrukciót a `nezetek`-től, és elvégzik a bemenet értelmezését, illetve meghívják a nézethez szükséges modulok inicializációját, destrukcióját és frissítését.

Rendre megjelenítik a főmenüt; magát a játékot; kezelik a dicsőséglistára a név megadását; kiírják az irányítás módját; és megjelenítik a dicsőséglistát.

mallokacio

a `mallokal` segédfüggvény itt kap helyet, meghívja a `malloc`ot, és ellenőrzi a sikerességét. A program összes `malloc` hívása ezen keresztül történik, tehát a `debugmalloc.h`-t is elegendő itt (a fejlécfájlban) includeolni. Emiatt az összes `free`-t használó modulban is includeolni kell a `mallokacio.h`-t, attól függetlenül, hogy mallokálnak-e.

vektor2, vektor2d, vektor2e

A d duplát, az e egészt, a 2 a két dimenziót jelöli, `vektor2` includeolja mindkét másikat és tartalmaz konvertáló függvényeket. A `vektor2d` és `vektor2e` modulok a típust, és standard vektorműveleteiket tartalmazzák.

lancolt_lista

A fő adatszerkezetért és függvényeiért felel, az ellenségek, a tornyok és a lövedékek használják. Emiatt általánosabb megvalósítás kellett, ahol a listaelem egy `void *`-t tárol.

Adat letörléshez mindenképp a `letorol` (vagy a `kiurit`) használandó a mutatók helyes beállítása és a kettő free hívás miatt. A `hozzafuz` függvény által kapott pointert is felszabadítják tehát a letörlő függvények.

A listaelem eltárolja az előzőt és a következőt is, mert így tud egyszerűbben letörölni egyetlen pointerrel átvett listaelemet.

Azért láncolt listára esett a választás, mert gyakran kell a lista közepéről eltüntetni elemeket (véletlenszerű ellenség meghal, lövedék talál, stb.), ami egy tömbnél nehézkes.

A listának nincsenek strázsái, ez ugyan egyes helyeken megnehezíti a speciális esetek kezelését, de cserébe a listát nem kell inicializálni: a `{NULL, NULL}` állapot is érvényes, és a lista ürességét jelzi. Emiatt a tornyoknak, lövedékeknek és az ellenségeknek sincs szüksége inicializáló függvényre.

ablak

Az SDL inicializációjáért, és a megjelenítő odaadásáért felel. Mivel nagyon sok függvényhívás történik addig, ameddig például egy ellenség kirajzolásáig eljut a kód, feleslegesen olvashatatlanná tenné, hogyha mindegyik függvénynél paraméter lenne a megjelenítő. Ehelyett a `szerez_megjelenito()` függvény használatával az bármikor lekérdezhető, az pedig a modul statikus globális változójával könnyen vissza tud térni, így azt elkerülve, hogy a megjelenítő indokolatlanul kívülről megváltoztatható legyen.

Létre tud hozni hardveres, függőleges szinkronizáció nélküli hardveres, és szoftveres megjelenítőt is.

keprajzolo

Beolvassa a képeket, adataikat, és a Kep felsorolással megadott bármely képet ki tudja rajzolni a képernyőre. Kétféle kirajzoló függvény van, az egyik egész képpontokra, a másik törtekre is. Erre azért van szükség, mert az SDL alacsony felbontásról nagyítja az ablakot, és így a valóságban a megadott tört pixelértékek egész értékeknek is megfelelhetnek, ezáltal az ellenségek és a lövedékek mozgása sokkal simább lesz. A KépInformáció típus egy kép adatait tárolja, ebből leggyakrabban a méret és az animáció képkocka száma érdekes a többi modul számára.

tornyok, ellensegek, lovedekek

A `*fajta` (pl.: Ellensegfajta) kívülről is elérhető felsorolást, a `*tipus` (pl.: Toronytipus) csak belülről elérhető további adatokat jelöl (pl.: lövedékfajta (toronynál), gyereke (ellenségnél), kezdősebesség (lövedéknél)). Ezentúl van egy `*tipusok` (pl.: Lovedektipusok) nevű konstans tömb, ami a felsorolásban szereplő fajták különböző adatait tárolja, `*fajtá`-val indexelhető.

Az entitások Ellenseg, Torony vagy Lovedek típusúak, amiben a konkrét pozíciójuk, sebességük, stb. illetve a `*tipus`ukra egy mutató van. Ezek egy statikus, globális láncolt listában foglalnak helyet, és a rajtuk végzendő műveleteknél a listán kell végigszaladni.

A statikus globális megoldásra az ablaknál (megjelenítő) leírtakhoz hasonló okokból esett a választás, illetve ha lenne egy mindentegybe játék struktúránk (amiben többek közt az ellenségek, lövedékek, tornyok listái is szerepelnek), akkor annak mindenhová adogatásával kevésbé lenne átlátható a program, mert nem lehetne kívülről tudni, hogy az adott függvény mégis a struktúra mely részeit változtatja. Így egyből látható, hogyha egy függvény pl. nem az ellenségek modul függvénye, akkor az biztosan nem nyúl az ellenségekhez.

Ezek a modulok erőteljesen hívogatják egymás függvényeit, lövések tesztelésére, célzás kiszámítására, ellenségek észlelésére, stb.

palya

A pálya beolvasásáért, kirajzolásáért, az út felismeréséért felel. Képes megmondani, hogy egy `double megtett_ut` milyen koordinátán helyezkedik el az úton.

jatek_ff

Csak a játék felhasználói felületének kirajzolásáért, és az ahhoz szükséges dolgokért felel, a bemenetet a `jatek_nezet` értelmezi.

hullamok

A tetszőlegesen hosszú és tetszőlegesen sok hullám beolvasásáért és az ellenségek megfelelő időpontban idézéséért felel. Képes a hullámokat végteleníteni.

elet, penz

Az élet, pénz és árazás területeiért felelnek.

dicsoseglista

Be tudja olvasni a fájlból a dicsőséglistát, azt ki tudja rajzolni, és hozzá tud adni egy új elemet, a fájlba visszamentéssel együtt. Előkerülő hibákat képes lekezelni.

modositott_szovegtipus.h

A `stringRGBA` által rajzolandó karakterek nyers adatait tárolja, főprogram olvassa be. Módosított Codepage-437 kódolású, hogy az összes magyar kis- és nagy ékezetes betűt képes legyen megjeleníteni. A módosítások a fájl tetején olvashatók.

Felhasználói dokumentáció

Program indítása

Ha a program valamiért nem indul, érdemes parancssorból elindítani, és figyelni a szabványos hibakimenetet. Ha a megjelenítővel van probléma, érdemes a `-s` argumentummal szoftveres megjelenítéssel próbálkozni.

Játékmenet, menük

Lásd: pontosított specifikáció.

Olvasott és írott szövegfájlok formátuma

- UTF-8 formátumúak (bár ASCII-n kívül nem volna szabad más karakternek lennie bennük)
- a dicsőséglistát kivéve, az Code page 437 kódolású mert a `stringRGBA` azt használja
- nem kezdődnek BOM-mal (bájtsorrend jelölő karakterekkel)
- sortöréssel végződnek
- lehetőség szerint LF-et használnak sortöréshez, bár ha minden igaz, a C könyvtár megoldja a CRLF kezelését is

Hibakódok

A program visszatérési értékei:

- -1: hibás parancssori argumentumok, vagy segítségkérés (-h)
- 0: helyes működés, bezárták a programot
- 1: SDL inicializációjának valamelyik része nem sikerült
- 2: dinamikus memóriaallokáció nem sikerült
- 3: valamelyik feltétlenül szükséges fájlt nem sikerült megnyitni
- 4: valamelyik fájl tartalmában olyan hiba van, amivel nem lehet a programot folytatni

Parancssori argumentumok

`./PinceTD -opciok` formátumban kell megadni, ahol `o`, `p`, `c`, `i`, `o` és `k` közül az érvényes kapcsolók értelmezve lesznek. `./PinceTD -h` segítségével láthatók a különböző kapcsolók, és az, hogy mit kapcsolnak. Ezekkel állítható be a megjelenítő: hardveres/szoftveres, legyen-e függőleges szinkronizáció (vsync) is.