S.U.M.S.U.M.

Super Unusual Medieval Strategic Ultimate Mission

Programozói dokumentáció

1. Bevezetés

A játékom (fő része) programozói szempontból egy fiktív világ szimulálása és a szereplőinek irányítása: a falvaknak minden egyes pillanatban ki kell számítani az állapotát; a számítógép által irányított falvakon le kell futtatni a mesterséges intelligencia algoritmusát; a menetelő zászlóaljaknak ki kell számítanom a pozícióját; a harcok kimeneteleit is meg kell állapítani; stb.

Ezen felül a felhasználó által generált eseményeket is néznem kell, azokra reagálni kell, megfelelő algoritmusokat lefuttatni, stb.

És mivel a program rendelkezik menüvel, ott is figyelnem kell a program eseményeit, kirajzolni a megváltozott menüt, stb.

Ebből is látszik, hogy a programom eléggé összetett, ezért fontos volt, hogy könnyen bővíthető és megfelelően strukturált kódot írjak. Így a programomat állapotokra, azokat külön fájlokra, és még néhány egyéb fájlra bontottam.

2. `globals.c` és `globals.h`

Ez a két fájl tárolja az összes saját típus definícióját, amit használok a programban (a főmenü struktúráját, a falvak információit tároló struktúrát, stb.), a globális függvényeket, mint az állapotváltás függvényét, és ezen kívül még ebben hozom létre azt a néhány változót (struktúrát), amit az egész programban használok.

A játékomnak nem kell tudnia egyszerre több világot szimulálni, egyszerre több ablakkal rendelkezni, vagy több főmenüt tartalmazni, ezért ezekhez praktikusabb volt globális változókat használni, amiket nem kell minden függvénynek átadni, mivel mindegyik mindenhol eléri.

A fontosabb típusok, függvények:

2.1. `enum GLOBALS {...};`

Ebben az enumerációban globális konstansokat tárolok, mint például egy katona nyersanyagköltsége vagy egy zászlóalj mérete. Ez a program könnyű konfigurálása miatt lett létrehozva.

2.2. `enum PROGRAM_STATES {...};`

Ebben az enumerációban tárolom le a játék lehetséges állapotait, amit állapotváltáshoz használók.

2.3. `struct PROGRAM_PROPERTIES {...};`

Ez a struktúra tárolja a program főbb tulajdonságait: a videokimenetet; az utolsó esemény struktúráját; a fő algoritmus utolsó lefutásának idejét, és a lefutások számának másodpercenkénti maximumát; a kurzor helyét; a program állapotát; és a betöltött betűtípusokat.

2.4. `struct MAIN_MENU_PROPERTIES {...};`

Ez a struktúra tárolja a főmenü összes tulajdonságát: a játékos által begépelt nevet; egyes menüpontok kiválasztott értékeit; és az aktív (kijelölt) menüpontot. Az utóbbi megvalósítása miatt a menü egy saját állapotgéppel rendelkezik (és a menüpontok is).

2.5. `struct IMAGES {...};`

Ebbe a struktúrába töltöm be az összes képet a program elindulásakor, amiket utána bárhonnét elérek.

2.6. `struct TOWN {...};`

Ez tárolja egy adott falu összes tulajdonságát: pozíció; szín; parasztok, nyersanyagok, katonák száma; és mivel a falukat láncolt listában tárolom, a struktúra tartalmaz egy pointert a lista következő elemére.
Amúgy ez a lista egy egyszeresen linkelt, elején strázsás lista.

2.7. `struct UNIT {...};`

Ez a struktúra tárolja egy adott zászlóalj tulajdonságait: honnét megy; hova megy; milyen a színe; mikor indult; és mikor fog megérkezni.
És mivel a zászlóaljakat egy kétszeresen linkelt, mindkét végén strázsás listában tárolom, minden egyes struktúra tartalmaz két pointert az előző és a következő zászlóaljra.

2.8. `sturct MATCH_PROPERTIES {...};`

Ez a struktúra tartalmazza a játszma összes tulajdonságát: mikor kezdődött; honnét nézi a játékos a világot; hány ellenség van; milyen nehézség lett kiválasztva; mekkora a minimális távolság az egyes faluk között (ezt a főmenüben a `Map size`-al lehet állítani); mik a kezdeti értékek (paraszt/nyersanyag/katona); ez a struktúra tárolja a falvak listájának kezdeti strázsáját; a zászlóaljak listájának mindkét strázsájának címét; a játékos színét, nevét; és a játékos által kiválasztott (irányítandó) falut és a kiválasztott ellenséges falut.

2.9. `void PROGRAM_SWITCH_STATE(PROGRAM_STATES STATE);`

Ez a függvény a program állapotának váltására való, mivel elmenti, hogy milyen állapotra akarunk váltani.
Ezek után a program megnézi, hogy akarunk-e állapotot váltani, és ha igen, akkor meghívja az új állapot inicializáló függvényét, az előző állapot un-inicializáló függvényét, majd beállítja az új állapotot.

3. `game_methods.c` és `game_methods.h`

Ez a két fájl az alapvető játékbeli algoritmusokat tartalmazza, mint a mesterséges intelligencia, paraszt/katona készítése egy adott faluban, zászlóalj kiküldése egyik faluból a másikba, stb.

- `void TOWN_CREATE_PEASANT(TOWN *this);`
  Ez a függvény egy adott faluban létrehoz egy parasztot, ha a falu elég nyersanyaggal rendelkezik. A falu memóriacímét paraméterként kapja.

- `void TOWN_CREATE_WARRIOR(TOWN *this);`
  Hasonlóan az előző függvényhez, ez is egy falut manipulál, de ez egy katonát készít. Paramétere szintén egy falu memóriacíme.

- `void UNIT_CREATE(TOWN *from, TOWN *to);`
  Ez a függvény kiküld egy zászlóaljnyi katonát a `from` faluból a `to` faluba, elmenti az indulás idejét, és kiszámolja a megérkezés idejét.
  A kiküldés gyakorlati megvalósítása pedig a zászlóaljakat tároló listába való fűzést jelenti.

- `int PLAYER_VICTORY();`
  Ez a függvény megnézi, hogy a játékos nyert-e: ha igen, 1-t ad vissza, minden más esetben 0-t.

- `int PLAYER_DEFEAT();`
  Ez a függvény az előző ellenkezőjét csinálja, azt nézi, hogy a játékos vesztett-e. Ha igen, 1-t ad vissza, ha nem, 0-t.

- `void AI(TOWN *this);`
  Ez a függvény tartalmazza a mesterséges intelligencia algoritmusát. A játék fő ciklusában minden falura meg van hívva, kivéve a játékos saját faluira.

- `void TTF_RenderText_Outline(SDL_Surface *dst, char *string, TTF_Font *font, int X, int Y, SDL_Color Ctxt, SDL_Color Cout);`
  Ez a függvény kilóg a sorból, ugyanis ez csak rajzol, megadott szöveget megadott képre megadott helyre megadott színnel és megadott körvonallal. A sorminta elkerülésének érdekében lett létrehozva, csak a faluk tartalmának kirajzolásánál használom. (parasztok, nyersanyagok és katonák száma)

4. A játék főbb állapotai és azok fájljai

Mivel a játék több részből áll, könnyen több állapotra bontható. Az állapotok a következők: főmenü, játszma közben, a játékos nyert, a játékos vesztett, dicsőséglista, kilépés. Minden állapothoz két fájl tartozik, egy `.c` és egy `.h`, és minden állapot 3 függvénnyel rendelkezik: `inicializáló`, `fő algoritmust tartalmazó` és `un-inicializáló`. Ezen belül pedig a fő algoritmusokat tartalmazó függvények is 3 külön részre bonthatóak: `eseményfeldolgozó`, `számításokat végző` és `kirajzoló rész`. (ám ezek a forrásban nincsenek külön függvényekre bontva, viszont el vannak tagolva egymástól.)

4.1. `state_main_menu.c` és `state_main_menu.h`

Ez a két fájl írja le a főmenü alapvető működését.

- `void STATE_MAIN_MENU_INIT();`
  Ez a függvény inicializálja a menüt, de mivel a menü állapotát tároló függvény már a program elindulásakor létrejön, így nem kell inicializálni semmit, így a függvény üres.

- `void STATE_MAIN_MENU_LOOP();`
  Ez a függvény felel a menü működéséért. Az eseményfeldolgozó rész feldolgozza a billentyűleütéseket, ezáltal lehet lépkedni a menüpontok között, választani az értékek között és nevet beírni. Szintén ennek a résznek köszönhető, hogy a játszma is elindítható, mivel itt hívódik meg a program állapotát váltó függvény is.
  Ha pedig változás történt a menüben, akkor újra kirajzolja.

- `void STATE_MAIN_MENU_UNINIT();`
  Mivel nincs semmi feltakarítandó vagy visszaállítandó érték a menüben, ezért ez a függvény is üres.

4.2. `state_playing.c` és `state_playing.h`

Ez a 2 fájl felel a játszma alapvető dinamikájáért, nem hiába ez az állapot a legösszetettebb és a leghosszabb is.

- `void STATE_PLAYING_INIT();`
  Ez a függvény előkészíti a játszmát: lefoglalja a láncolt listák strázsáit; létrehozza a falvakat; beállítja a kezdeti értékeket; és megvizsgálja a nézőpont szélsőértékeit. (emiatt nem lehet túl messzire menni a térképen.)

- `void STATE_PLAYING_UNINIT();`
  Ez a függvény a játszma utáni „takarítást” végzi el: felszabadítja a láncolt listákat. Más feladata nincs.

- `void STATE_PLAYING_LOOP();`
  Ez a függvény a játék lelke: feldolgozza a felhasználó által generált eseményeket; reagál rájuk; kiszámolja a falvak nyersanyagmennyiségét;

megvizsgálja az összes menetelő zászlóaljat, amik ha beértek, akkor kiszámolja a harcok kimenetelét is; falufoglalás esetén megnézi, hogy a játékos nyert/vesztett-e; lefuttatja a mesterséges intelligencia algoritmusát a számítógép által vezérelt falvakon; és végül kirajzolja a világot a képernyőre.

4.3. `state_victory.c` és `state_victory.h`

Ez a 2 fájl a nyerési állapotot írja le. Akkor lép a program ebbe az állapotba, amikor a játékos megnyerte a játszmát, azaz sikeresen elfoglalta az összes falut.

- `void STATE_VICTORY_INIT();`
  Ez a függvény inicializálja az állapotot: kiszámolja, hogy hány pontot ért el a felhasználó; beolvassa az eddigi dicsőséglistát; megnézi, hogy be kell-e szúrni a felhasználó pontszámát a listába, és ha igen, beszúrja, majd kiírja az új dicsőséglistát; és végül kiírja a képernyőre a `Victory!` szöveget, tudatva a felhasználóval, hogy nyert.

- `void STATE_VICTORY_LOOP();`
  Ez a függvény nem csinál semmit, csak eseményre vár, és ha az esemény egy `ENTER` leütése, akkor a program állapotot vált a főmenü állapotára.

- `void STATE_VICTORY_UNINIT();`
  Ez a függvény nem csinál semmit, mivel nincs mit „feltakarítani”/beállítani az állapotból kilépéskor.

4.4. `state_defeat.c` és `state_defeat.h`

Ez a 2 fájl a vesztett állapotot írja le, amikor a felhasználó összes faluját elfoglalták és nem rendelkezik egy zászlóaljjal sem.

- `void STATE_DEFEAT_INIT();`
  Ez a függvény a vesztett állapotot inicializálja: kiírja a képernyőre a `You have been defeated!` Szöveget.

- `void STATE_DEFEAT_LOOP();`
  Ennek a függvénynek csak eseményfeldolgozó szerepe van: eseményre vár, és ha az pont egy `ENTER`-leütés, akkor visszalép a főmenübe. (állapotot vált)

- `void STATE_DEFEAT_UNINIT();`
  Ez a függvény üres, mivel nincs mit feltakarítani/visszaállítani az állapotból kilépéskor.

4.5. `state_view_highscores.c` és `state_view_highscores.h`

Ez a 2 fájl a dicsőséglista megjelenítéséért felel.

- `void STATE_VIEW_HIGHSCORES_INIT();`
  Ez a függvény inicializálja az állapotot: elsötétíti a menü képét, beolvassa a dicsőséglistát, majd kilistázza a képernyőre.

- `void STATE_VIEW_HIGHSCORES_UNINIT();`
  Ez az függvény is üres, mivel nincs mit feltakarítani/beállítani.

- `void STATE_VIEW_HIGHSCORES_LOOP();`
  Ez a függvény szintén csak eseményre vár, és ha az egy `ESC`-leütés, akkor a program visszalép a főmenübe.