Programozói és felhasználói dokumentáció

NHF – Snake – Molnár Martin – SQ997J – 2013.12.08 11:54

A játék (felhasználói dokumentáció):

Egyszerű, konvencionális kígyós játék, melyben lehetőség van többjátékos módra, illetve a pontok alapján készített ranglista megtekintésére. Egyjátékos módban a nyilak segítségével irányítható a kígyó, többjátékos módban nyilak+WASD. Mindig a legjobb öt pont van eltárolva, játék végén ezek felülíródnak, amennyiben a játékos(ok) jobb eredményt értek el a ranglistában levőknél.

Adatszerkezet:

A megfelelő memóriakezelés érdekében duplán láncolt lista szerkezetű a kígyó, más adatszerkezettel csak körülményesen lehetett volna megvalósítani a függvényeket.

Feltűnhet, hogy viszonylag kevés változó van a struktúrákon belül. Ennek az oka, hogy a kígyó minden egységének véletlenszerűen számolt színe van, az egér színe pedig konstans. A struktúrákon felül a játék lényeges része egy öt sorból álló txt fájl is, ami a program gyökerében kapott helyet, ebben tárolódik a legjobb öt pontszám. A fájl fixen öt sorból áll, külső beavatkozás nélkül ez nem is változik, a program írása ezt a tényt figyelembe véve történt, tehát az ideiglenes függvényen belül változók, melyek a lista elemeit tárolják nem dinamikusan foglaltak, mivel felesleges. A játék az említetteken kívül más adatszerkezetet nem tartalmaz.

```
typedef enum iranyok{
  fel,le,jobbra,balra

}iranyok;

typedef struct egyseg{

        iranyok irany
        int x;
        int y;
        struct egyseg *kovetkezo;
         struct egyseg *elozo;

}egyseg;

typedef struct eger{
        int x;
         int y;
}eger;
```

A program működését vezérlő fő függvények:

A program modulokra oszlik, melyek a következők:

```
foglal.c
rajzol.c
menu.c
egyeb.c
global.h
```

A global headerfile-on belül van definiálva az összes függvény, melyet a program használ, kommentekkel felbontva. A függvények bemutatását abban a sorrendben fogom végezni többé-kevésbé, ahogy a fordító értelmezné.

*int menu(SDL_Surface *screen)*

SDL Timer segítségével egy ciklusban folyomatosan kirajzoltatja *a void menu_rajzol(SDL_Surface *screen, int menupont)* függvénnyel a menüpontokat. Az aktív menüpontot egy int változó értékének módosításával adja át, melyet a billentyű lenyomásának függvényében változtat. A menu visszatérési értéke alapján történik a memóriafoglalás.

*egyseg * fej_foglal(void)*

A függvény egy egysegnyi memóriaterületet allokál, majd a *int fej_inicializal(egyseg *feje)* függvényt meghívva inicalizálja a fejet.

int pont_rajzol(screen)

Ha a menu függvény visszatérési értéke 3, ez a függvény hívódik meg. Az SDL_ttf könyvtár segítségével a pontok.txt fájlból beolvasott adatokat írja ki.

*void eger_rajzol(SDL_Surface *screen, eger *e,egyseg *eleje)*

A függvény először meghívja az *eger_inicializal(e,eleje)* függvényt, majd a kapott koordináták helyén négyzetet rajzol. Ennek a függvénynek van párja, mely a *void eger_rajzol2(SDL_Surface *screen, eger *e,egyseg *eleje)*, ennek a működési elve ugyanaz csak 3 paramétert kap, a két játékos mód miatt.

*void leptet(SDL_Surface *screen , egyseg *eleje)*

A függvény a paraméterként kapott kígyó eleje mutató segítségével törli az éppen kirajzolt kígyót a *void torol(SDL_Surface *screen, egyseg *eleje)* függvény meghívásával, majd a kígyó egységeinek új irányt ad (az adott egység az előző irányát kapja meg), majd ezt a *void rajzol(SDL_Surface *screen , egyseg *eleje)* függvény segítségével kirajzolja a képernyőre.

*int utkozes_eger(SDL_Surface *screen, eger *e, egyseg *eleje);*
*int utkozes_fal(egyseg *eleje);*
*int utkozes_test(egyseg *eleje);*

Ezek a függvények lényegében ugyanazt csinálják eltérő kondíciókkal, az egér ütközés értelemszerűen a kígyó fej egységének koordinátáit vizsgálva tér vissza igaz vagy hamis értékkel, falhoz való ütközés szintén így működik, az utkozes_test függvény viszont érdekesebb, de a végén az is egy if-el zár és az alapján kapjuk a visszatérési értéket.

```
int utkozes_test(egyseg *eleje)
{
   if(eleje->kovetkezo!=NULL){//a saját fejével nem tud ütközni
      egyseg *mozgo=eleje->kovetkezo;

      while(mozgo->kovetkezo!=NULL)
      {
         int mozgox=mozgo->x;
         int mozgoy=mozgo->y;
         int kigyox=eleje->x;
         int kigyoy=eleje->y;
         if(kigyox==mozgox&&kigyoy==mozgoy)
         {
            return 1;
         }
         else
         {
            mozgo=mozgo->kovetkezo;
         }
      }
   }
   return 0;
}
```

A kígyó egységein végigmegy egy ciklussal és, ha az adott elem koordinátái egyeznek a fej koordinátáival 1-el tér vissza. Ennek a három függvénynek van párja pl. *int utkozes_test2(egyseg *eleje)*, melyek kapnak még egy paramétert, a második kígyó mutatóját, de lényegében ugyanúgy működnek.

int pont_kezel(int pont1, int pont2)

Ez a függvény hívódik meg, ha a fent bemutatott függvények egyike igazra értékelődik ki. Megnyitja a pontok.txt-t, abból beolvassa az adatokat, melyet egy int típusú tömbben tárol. A paraméterként kapott értékeket hozzáfűzi a tömbhöz, majd a C beépített qsort algoritmusával rendezi azt, ezután visszaírja a fájlba a legjobb öt eredményt, tehát a tömb utolsó két elemét nem.

*int kigyo_felszabadit(egyseg *kigyo);*

Az SDL_Quit előtt hívódik, a kígyó eleje mutató segítségével végigiterál a láncolt listán és minden egyes elemét felszabadítja, ha ez megtörtént visszatér 1-el.

Zene:

A játék eredetileg tartalmazott volna zenét, de a fájlméret korlát miatt ezt kihagytam a beadott verzióból. A forráskódban viszont ott van az összes zenével kapcsolatos kód kikommentelve, hogy azért lehessen látni milyen lenne egyébként.

Az összes függvény listája:

```
//foglal.c
int eger_inicializal(eger *e,egyseg *eleje);
int eger_inicializal2(eger *e,egyseg *kigyo, egyseg *kigyo2);
int fej_inicializal(egyseg *feje);
int egyseg_inicializal(egyseg *test);
egyseg * fej_foglal(void);
int kigyo_novel(egyseg *kigyo);
int kigyo_felszabadit(egyseg *kigyo);

//rajzol.c
void eger_rajzol(SDL_Surface *screen, eger *e,egyseg *eleje);
void eger_rajzol2(SDL_Surface *screen,eger *egerke,egyseg *kigyo,egyseg *kigyo2);
void eger_torol(SDL_Surface *screen, eger *e);
void rajzol(SDL_Surface *screen , egyseg *eleje);
void torol(SDL_Surface *screen, egyseg *eleje);
void leptet(SDL_Surface *screen , egyseg *eleje);
int rand_szam_RGB(void);
int pont_rajzol(SDL_Surface *screen);

//menu.c
int menu(SDL_Surface *screen);
void menu_rajzol(SDL_Surface *screen, int menupont);

//egyeb.c
Uint32 idozit(Uint32 ms, void *param);
int utkozes_eger2(SDL_Surface *screen, eger *e, egyseg *kigyo,egyseg *kigyo2);
int utkozes_eger(SDL_Surface *screen, eger *e, egyseg *eleje);
int utkozes_fal(egyseg *eleje);
int utkozes_test(egyseg *eleje);
int utkozes_test2(egyseg *kigyo,egyseg *kigyo2);
int pont_kezel(int pont1, int pont2);
int melyik_a_nagyobb( const void *a, const void *b);
```