
HF IV
„globs”

Specifikáció

- http://ingyenesjatek.hu/globs.html linken elérhető játékot kell megvalósítani konzolos környezetben
  - ***Menükezelés nem kell***, csak maga a játékmenet
    - Lépés számlálás kell
    - Szintléptetés és az azzal kapcsolatos játéktér növekedés kell
    - Pontszámítás opcionális (bónusz %-ért)
  - Vezérlés az 1-6 billentyűkkel (mint ahogy az eredeti játékban is)
  - A golyók helyett teli színes téglalapok legyenek
    - Space (‘ ‘) karakter + háttérszín

Adatszerkezet kialakítás

- Mivel a konzolról nincs (egyszerű) lehetőséged visszakérni hogy melyik pozícióban milyen szín áll, ezért ajánlom, hogy egy általad kialakított 2 dimenziós tömbbe „számolj”, majd ha azzal végeztél az egész tömb tartalmát rakd ki a képernyőre

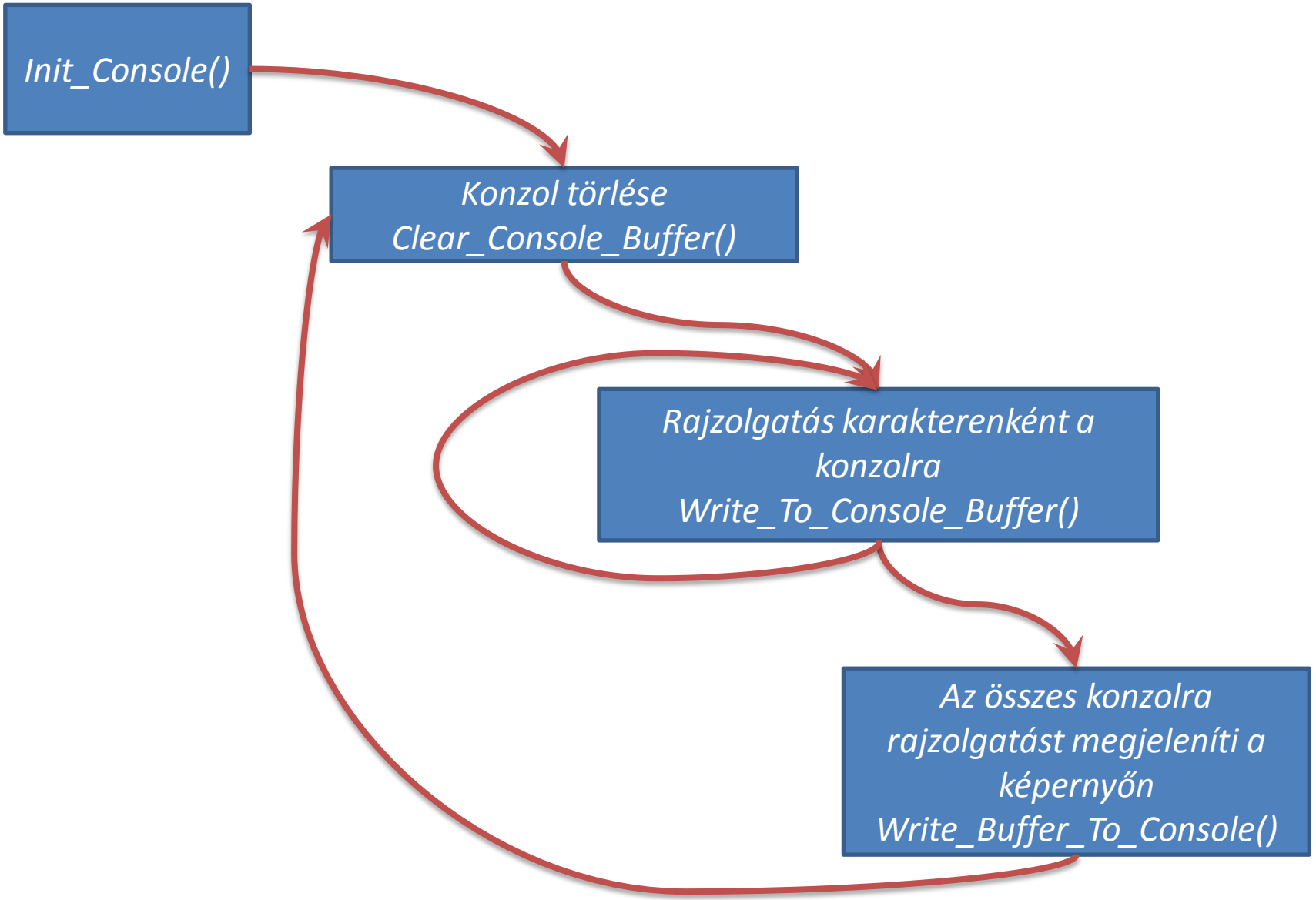
Konzolra rajzolás

- „Dropbox”\SZFe_2014\prog\c_strukturalt__Mintak\fastconsole könyvtárban találsz egy minta projektet ami az új „felturbósított” MyConsole ***modult*** tartalmazza amit használnod kell
  - emléxel? „modul = .c(pp) + .h(pp)”
  - mint régebben csak a MyConsole.h-t kellett a projekthez adni, most már a „c(pp)-jét” is

Konzolra rajzolás

- Nagysebességű konzolra rajzolás lépéseit megtalálod a minta projektben alaposan kommentelve, de azért itt tömören:
  - A main függvénye ***legeslegeslegelején*** meghívod a *Init_Console* függvényt
  - *Set_Console_Buffer_Sizes* fügyvénnyel beállítod mekkora ablakon akarsz dolgozni
    - ez nem a *játéktér,* hanem a konzolos ablakod mérete karakterekben
  - A konzolra rajzolás előtt törlöd az ablak „tartalmát” a *Clear_Console_Buffer* függvénnyel
  - Rajzolgatsz a „konzolra” *Write_To_Console_Buffer* függvénnyel, ahol meg kell adni:
    - X koordináta
      - szokásosan 0 a bal első oszlop a jobb oldali utolsó „buffer szélesség -1”
    - Y koordináta
      - szokásosan 0 a felső sor az alsó „buffer magasság-1”)
    - Színkód
      - A Set_Console_Text_Color függvénynél megszokott színkód pl.: FCIY | BCB
    - Karakterkód
      - Ahogy az szokásos pl.:‘*’
  - Ha megvagy a játéktér rajzolgatással, akkor meghívod a „Write_Buffer_To_Console” függvényt ami azonnal ki is rakja a képernyőre ami az utolsó képernyőtörlés (*Clear_Console_Buffer*) óta rajzolgattál
  - És kezded előről a rajzolgatást …

Konzolra rajzolás



Értékelés

- A játék azon logikája ami azt valósítja meg, hogy az elárasztott területet a kiválasztott golyó színével átszínezi (***területkitöltés***) , és hozzákapcsolja a kiválasztott golyó színével megegyező szomszédos golyókat összességében ***CSAK 20%-ot ÉR,*** mert ez a logika rekurzív jellegű és bár tanultátok, nem feltétlenül várom el ennek azonnali gyakorlatba ültetését
- A többi 80% arányosan oszlik a többi elvárt feladatrészre

Területkitöltés

- Az egyre nagyobb összefüggő színű területek beszínezésére alkalmas algoritmusok gyűjtő neve a „területkitöltő” (flood fill) algoritmusok, amelyekből ebben a feladatban a rekurzív változat tökéletesen megfelelő:
  - http://progalapm.elte.hu/downloads/eloadas/progalapm_ea5_graf2.pdf
  - http://tudasbazis.sulinet.hu/hu/informatika/informatika/informatika-9-12-evfolyam/elemi-grafikai-algoritmusok-pont-szakasz-kor-rajzolasa-alakzatok-szinezese-vagas/alakzatokat-kitolto-algoritmus
    - Vigyázz a függvényben lévő első „flood_fill” nem kell mert végtelen ciklus :D ---→

```
public void flood_fill(int x,int y)
{
    flood_fill(x,y);
    if (getpixel(x,y) == hatterszin)
        {
        WritePixel(x,y,szin);
        flood_fill(x+1,y);
        flood_fill(x-1,y);
        flood_fill(x,y+1);
        flood_fill(x,y-1);
        }
}
```

- Érdemes az algoritmust papíron kézzel átgondolni!