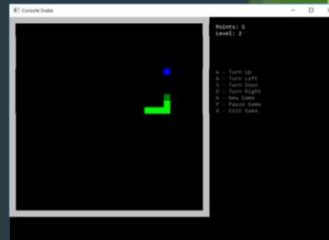




## Bemutató: Console Snake

- ▶ Szokásos kígyós játék, egyszerű megoldásokkal.
- ▶ Megjegyzésekkel és üres sorokkal együtt kb. 300 sor (könyvtár nélkül).
- ▶ C++11-ben íródott, így látszik a könyvtár használhatósága C++ nyelven is.
- ▶ Egyszerűsége miatt választottam.



Egy konzolos Snake játék kiválóan prezentálja, hogy milyen típusú feladatokat könnyít meg hatalmas mértékben a CML.

# Motiváció

## WinAPI megoldás a Microsoft leírásai alapján

```
#define PERR(bSuccess, api) { if(!bSuccess) printf("Ms:Error %d from %s \ on line %d\n", __FILE__, GetLastError(), api, __LINE__); }  
  
void cIs(HANDLE hConsole) {  
    COORD coordScreen = { 0, 0 };  
    BOOL bSuccess;  
    DWORD cCharsWritten;  
    CONSOLE_SCREEN_BUFFER_INFO csbi;  
    DWORD dwConSize;  
  
    bSuccess = GetConsoleScreenBufferInfo(hConsole, &csbi);  
    PERR(bSuccess, "GetConsoleScreenBufferInfo");  
    dwConSize = csbi.dwSize.X * csbi.dwSize.Y;  
    bSuccess = FillConsoleOutputCharacter(hConsole, (TCHAR) ' ', dwConSize, coordScreen, &cCharsWritten);  
    PERR(bSuccess, "FillConsoleOutputCharacter");  
    bSuccess = GetConsoleScreenBufferInfo(hConsole, &csbi);  
    PERR(bSuccess, "ConsoleScreenBufferInfo");  
    bSuccess = FillConsoleOutputAttribute(hConsole, csbi.wAttributes, dwConSize, coordScreen, &cCharsWritten);  
    PERR(bSuccess, "FillConsoleOutputAttribute");  
    bSuccess = SetConsoleCursorPosition(hConsole, coordScreen);  
    PERR(bSuccess, "SetConsoleCursorPosition");  
    return;  
}
```

↓  
ClearScreen();  
CML függvény

Kiemelendő, hogy mivel a CML a Windows API-n alapszik, nem tud többet, de egyszerűbbé, barátságosabbá teszi a használatát ezen a téren. Tehát a lényeg: kényelmetlen, de mindentudó API felhasználásával kényelmes, specializált API-t írni.

## A könyvtár használata

- ▶ MakeBuffer függvénnyel lefoglalunk egy puffert.
- ▶ Draw\* és Fill\* nevű függvényekkel rajzolunk a pufferbe.
- ▶ ClearBuffer függvénnyel ürítünk ki teljesen egy puffert. (Minden Pixel fekete lesz benne, szöveg nélkül.)
- ▶ PrintBuffer függvénnyel rajzoljuk ki a puffert.
- ▶ Használat után a FreeBuffer() függvénnyel szabadítjuk fel az előzőleg lefoglalt puffer memóriaterületét.

Rajzolás  
kódja

rajzolás,  
törlés

Konzolpuffer

végleges  
kimenet

Konzolablak

Itt látszik, mennyire hasonlít egy CML-es konzolban futó program alakja egy modern grafikai könyvtárban írt programéhoz, ahol szintén egy pufferbe írunk, és ezt rajzoltatjuk ki a rendszerrel.

## Tervezési célok

```
HANDLE cb = CreateConsoleScreenBuffer(  
    GENERIC_WRITE,  
    FILE_SHARE_WRITE,  
    NULL,  
    CONSOLE_TEXTMODE_BUFFER,  
    NULL);  
COORD size = { X, Y };  
SetConsoleBufferSize(cb, size);  
// és még csak lefoglaltunk egy írható puffert,  
// menteni pedig a C könyvtáron keresztül kéne, a  
// handle által kezelt területet, bájtonként
```

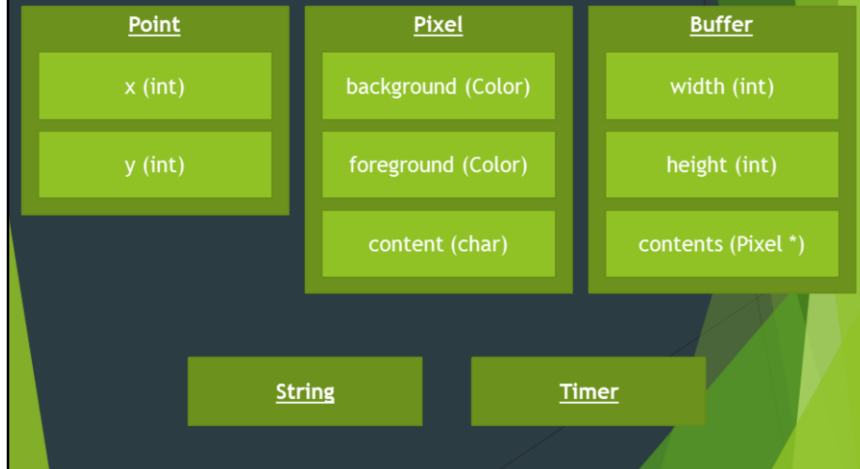
```
Buffer cb = MakeBuffer(X, Y);  
// ... pufferbe rajzolás  
PrintBuffer(cb);  
SaveBuffer("puffer.dat", cb);  
FreeBuffer(cb);
```

```
cb = LoadBuffer("puffer.dat");  
PrintBuffer(cb);  
FreeBuffer(cb);
```

A felső (WinAPI)  
kódrészlettel  
egyenértékű az  
alsó kódrészlet  
első sora!

Megjegyzendő, hogy a külső DLL-ek használatának elkerülése is cél volt, és hogy a Windows-os konzolpuffereket HELYETTESÍTI ez a könyvtár, nem pedig használja, valamint, hogy az egyszerűség volt a legelső cél, még ha ezzel némi teljesítménytől el is esünk (ami a mai gépeknél nem okoz gondot).

## A CML különleges típusai



Ez a dia szerény véleményem szerint öndokumentáló ☺.

## Néhány egyéb hasznos funkció

- ▶ SaveBuffer(), LoadBuffer()
- ▶ GetConsoleSize()
- ▶ StartTimer(), DeleteTimer()
- ▶ JumpToPosition()
- ▶ SetCaption()

A SaveBuffer() és LoadBuffer() parancsok pufferek fájlba mentését, illetve fájból betöltését segítik elő, a GetConsoleSize() visszaadja egy Point objektumban a konzolablak méretét (X = szélesség, Y = magasság), a StartTimer() és DeleteTimer() függvények Windows-os időzítőkkal hívnak meg egy bizonyos alakú függvényt, megadott időközönként, a JumpToPosition()-nal a konzolablakon belül ugorhatunk koordinátára, a SetCaption() pedig beállítja a konzolablak címsorában megjelenő szöveget.

## A CML jövője

- ▶ Aktív projekt, tehát folyamatosan bővülni fog.
- ▶ Szabadon használható és elérhető az interneten. (<http://github.com/denes-finth/cml>)

### További célok

- ▶ Teljesítménybeli javítások
- ▶ Több primitív alakzat kialakítása
- ▶ Nem primitív alakzatok kialakítása
- ▶ Belső hibakezelés kialakítása
- ▶ Képbe mentés függvény

Az előadáshoz képest bővült a dia a forráskód GitHub linkjével.