Óbudai Egyetem
Neuman János Infortamtikai Kar
Nappali tagozat

PerProg Féléves Beadandó

Név: Petrovics Gábor
Neptunkód: BP0G5T

Feladat címe: GameOfLife

Tartalom

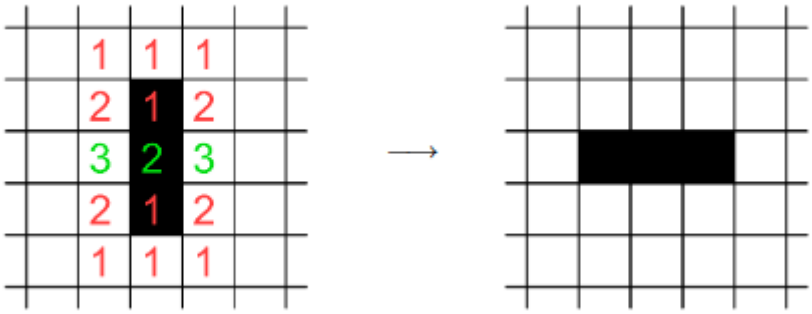
Beadandó terv

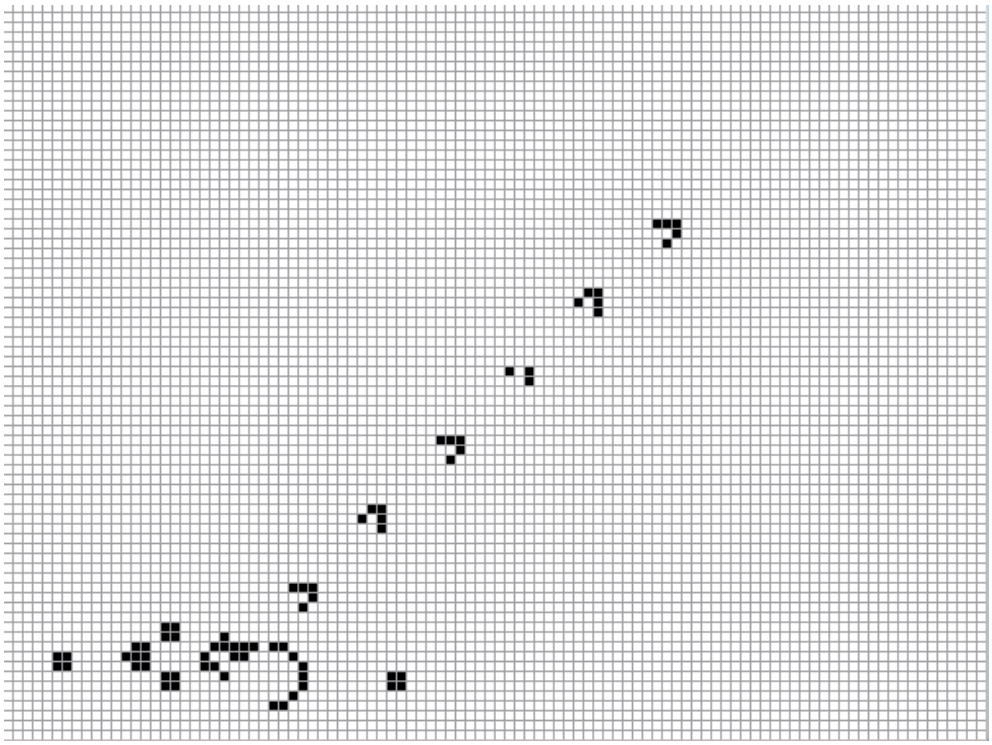
Én egy Celluláris automatát szeretnék csinálni más néven Game of Life, aminek lényege, hogy van egy élettér ami egy négyzetháló. Minden cellában egy sejt élhet, minden sejtet nyolc szomszédos cella vesz körül. A játék aktuális állapota az előzőbeli állapottól függ tehát a játék generációkra(iterációkra) van bontva amire a következő szabályok érvényesek:

- ha egy iterációban egy sejtnek kettő vagy három élő szomszédja van akkor a sejt a következő iterációban is élni fog.
- ha kettőnél kevesebb vagy háromnál több élő szomszédja van akkor a sejt elpusztul.
- ha egy üres cellának három élő sejt szomszédja van akkor egy új sejt jön létre az üres cellába.

Ezekből a szabályokból elég változó és érdekes alakzatok keletkezhetnek. Ez nagyban függ a kezdő állapottól.

Egy példa:





Specifikáció

Az élettér(négyzetháló) mérete mondjuk 600x600 legyen. A játék alapvetően nulla játékos, mivel igazából csak annyi a dolga, hogy a kezdő állapotot meghatározza tehát, hogy hol legyenek sejtek az elején. Egy másik lehetséges kezdő állapot meghatározó mód az egy Perlin zaj generátor lesz, ami feltölti a táblát random sejtekkel. Mivel a játék állapot változások vannak ezért ezt automatikusan egy(változhat) másodpercenként hajtja majd végre. A játék megjelenítését WPF-be akarom csinálni.

Megvalósítás

Az élettér mérete végül 160x160-as lett mivel a 600x600 elemet A WPF nagyon lassan jelenít meg. Az elején fel kell tölteni az életteret azaz a négyzethálót sejtekkel, amit úgy tehet meg a felhasználó, hogy az egér bal gombjával rá kattint az adott kis négyzetre és akkor ott lesznek sejtek ahova kattintott vagy egy AutoGenerate gombra kattintva le generálja a rendszer neki. A Start gombbal lehet elindítatni a köröket ami addig megy amíg a Stop gombra nem kattint.

A megvalósítás négy osztályból áll, a Table, BusinessLogic, Draw, PerlinGenerator.

Table

Ez szimbolizálja az életteret azaz a négyzetrácsot egy két dimenziós integer tömbben, ahol a 0 érték azt jelzi, hogy ott nincs sejt, az 1-es érték pedig azt, hogy van ott sejt.

Draw

A megjelenítési osztály ami egy FrameworkElement leszármazott. Itt történik a UI kezelése és megjelenítése. Itt kezelem az egér kattintás eseményét. Az élettér kirajzoltatásához egy RectAngleGeometry listát használok amit feltöltök inicializálásnál annyi elemmel amennyi a Grid mérete. Az újrarajzoltatást és a köröket egy DispatcherTimer időzítővel csinálom ami 50 ms-ként rajzoltat, illetve elindít egy újabb kört.

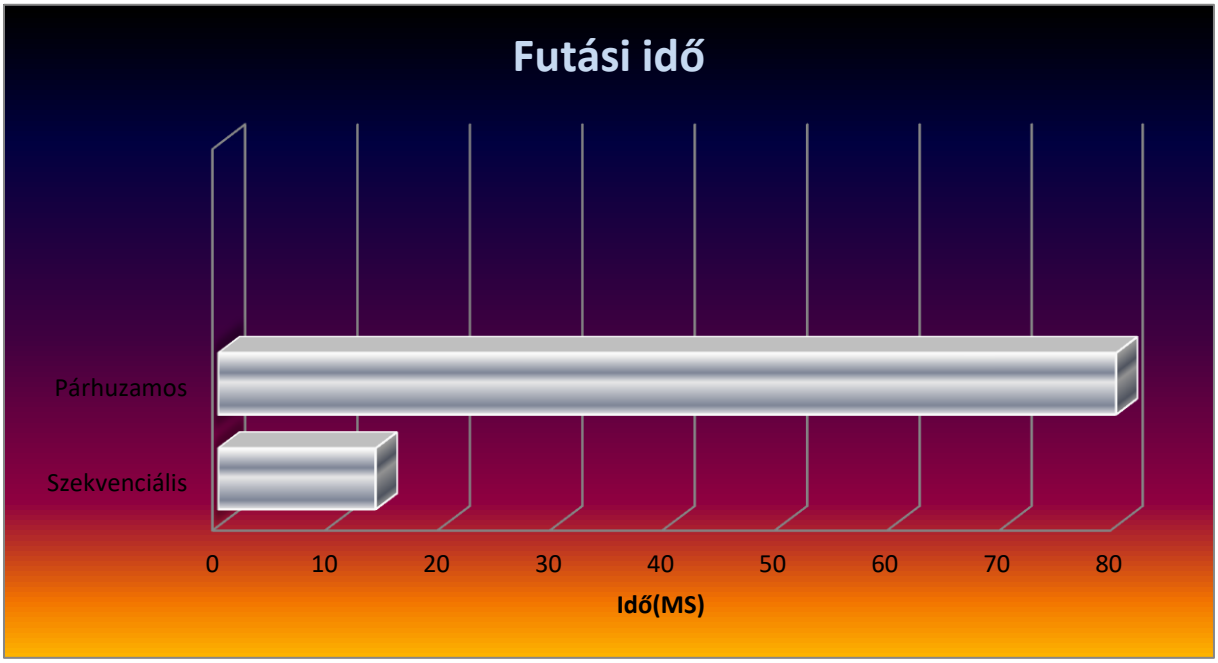
BusinessLogic

Itt történik a négyzetháló egyes kis négyzetei szomszédjainak a vizsgálása, illetve a kör indítása. Egy adott körben a sejtek szomszéd vizsgálatát úgy oldom meg, hogy a kör elején egy másolat táblát készítek és abba módosítom az eredeti tábla szerint a sejteket, tehát az integer tömb értékeit.

Párhuzamosítás:

I. Terv

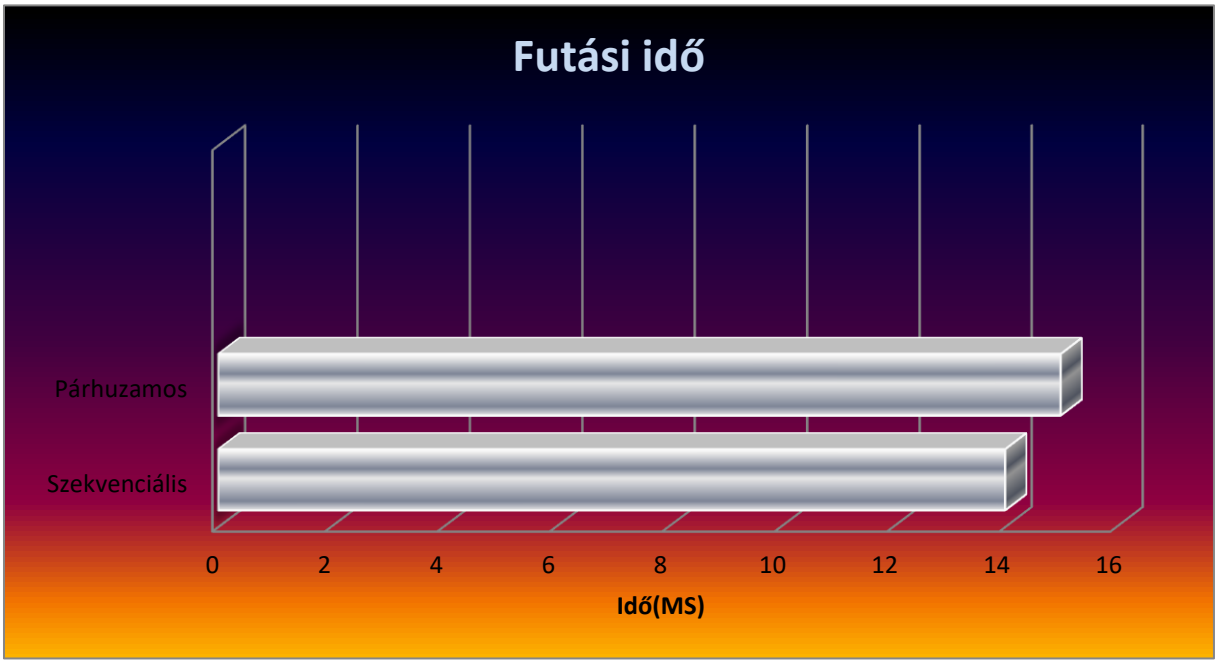
A négyzetek szomszéd vizsgálatát párhuzamosítottam. Kipróbáltam úgy, hogy minden egyes négyzetre egy külön taskot indítok így csak egy négyzet szomszédját kell vizsgálni egy tasknak de ez körülbelül tízszer lassabb volt ilyen tábla méretnél mint szekvenciálisan csinálnám.



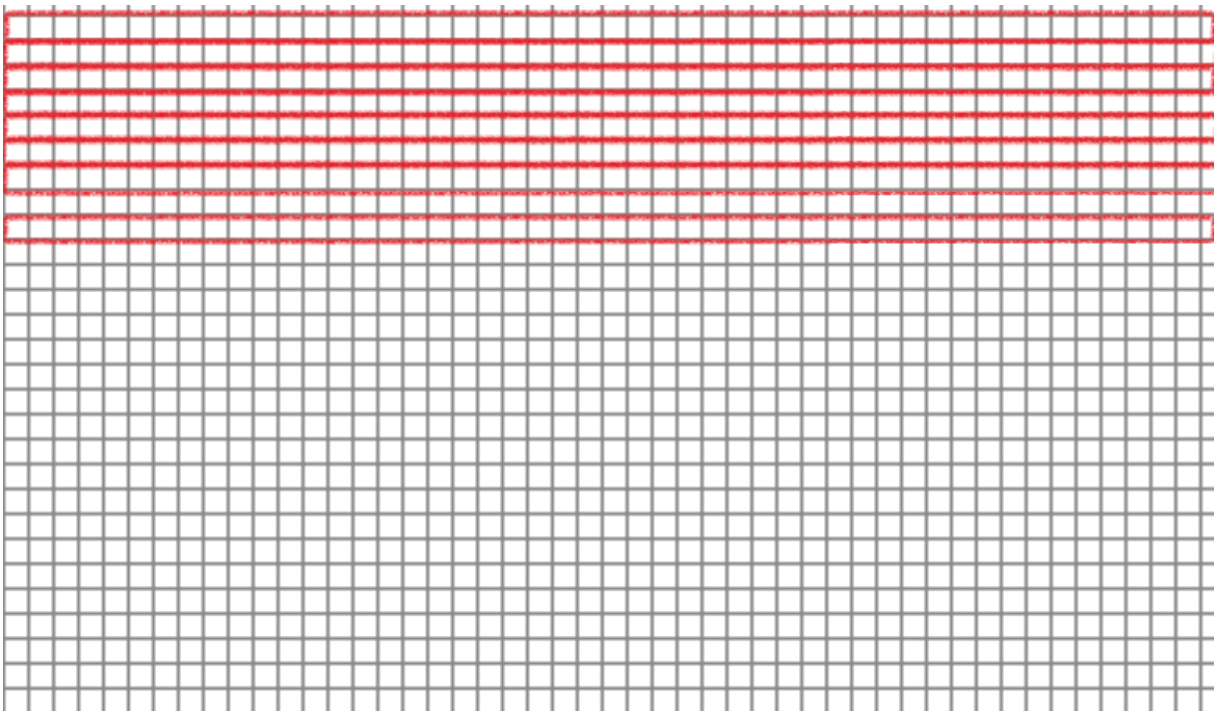
I. Terv futási ideje

II. Terv

A következő tervem az volt, hogy soronként vagy oszloponként vizsgálom, tehát minden sorra vagy oszlopra egy külön taskot indítok. Ez már jobb futási időt eredményezett de még mindig volt amikor lassabb volt mint a szekvenciális de volt amikor picivel gyorsabb is volt.



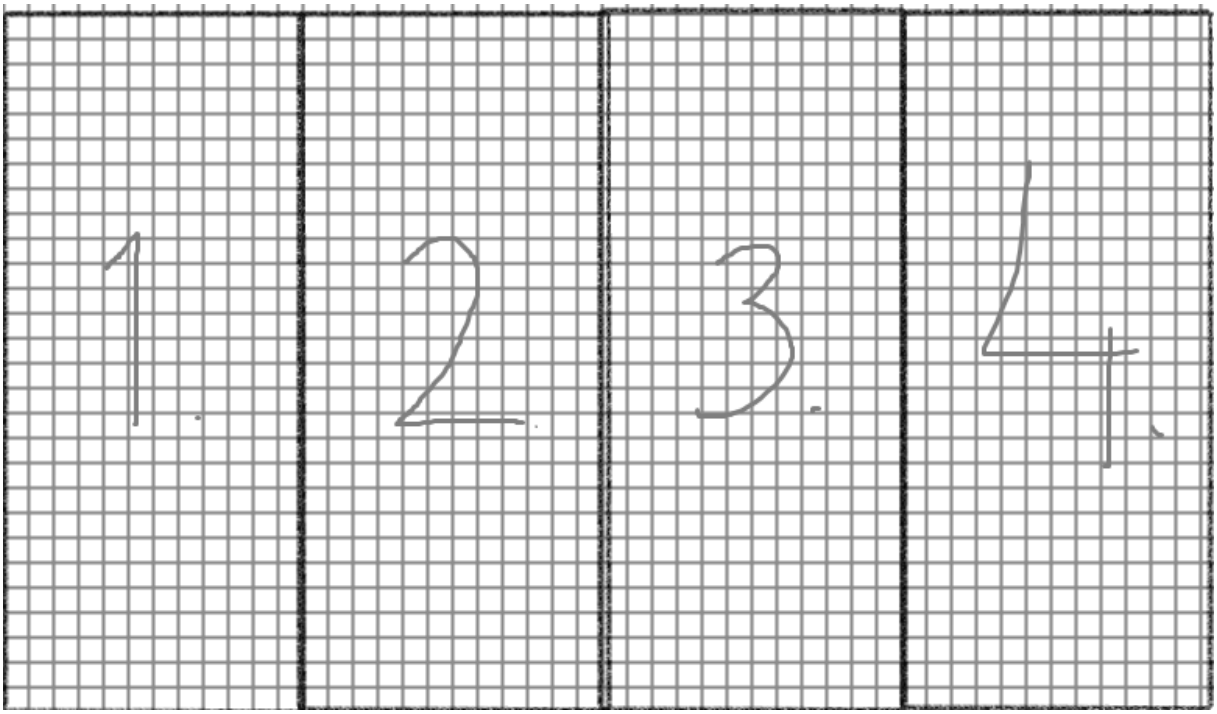
II. Terv futási ideje



Példa a soronkénti felosztásra

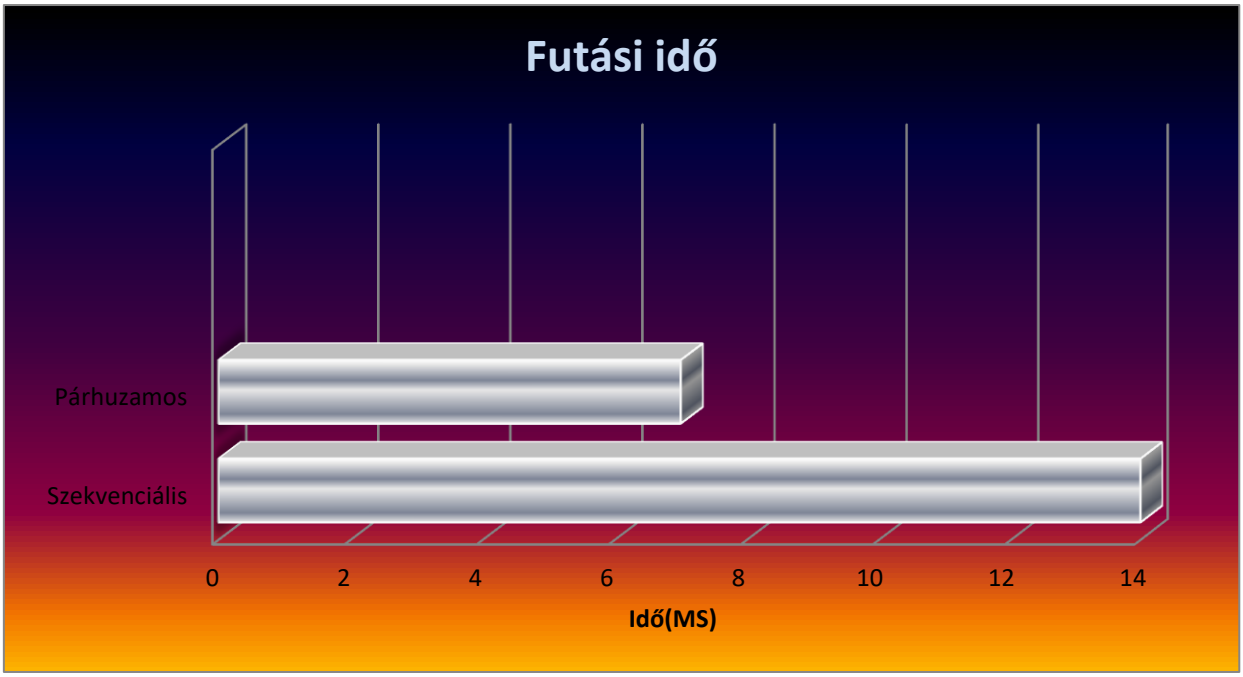
III. Terv

A végül az utolsó terv az volt, hogy annyi taskot indítok ahány szálas az adott CPU, Aztán felosztom a négyzetrácsot annyi kis téglalapra ahány szálas az adott CPU, így annyi taskot indítok ahány szálas az adott CPU, egy taskkal egy megadott téglalapon belül vizsgálom a kis négyzetek szomszédjait.



Példa Egy 4 szálas CPU-n

A felosztást úgy csináltam, hogy elosztottam a négyzet rács sorának a méretét az adott CPU-n szál számával ez megadta hogy mekkora legyen egy téglalap oldala a másik oldala akkora amekkora a négyzetrács oszlopainak mérete. Ezzel a tervel majdnem kétszeres gyorsaságot sikerült elérni.



III. Terv futási ideje