ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Lex Tamás József**
2022	Hallgató törzskönyvi száma:	**T/005724/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Lex Tamás József**
Törzskönyvi száma: T/005724/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Virtuális Petricsésze
Virtual Petridish

Intézményi konzulens: Dr. habil Szénási Sándor
Külső konzulens:

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Szoftvertervezés és –fejlesztés specializáció

A feladat

Tervezzen meg és implementáljon egy rendszert, amely képes sejtek növekedésének és szaporodásának szimulációjára! Vizsgálja meg a hasonló rendszerek működését és tekintse át a rendelkezésre álló (biológiai és informatikai) módszereket!
Az irodalomkutatás alapján készítse el a megvalósítandó program rendszertervét! Tervezze meg úgy a modellt, hogy az képes legyen több sejtből létrejött rendszerek szimulációjára is (mozgás, anyagáramlás, stb.). Implementálja az így megtervezett algoritmusokat és a szükséges felhasználói interfészt is! Tesztelje az elkészült rendszert, és értékelje az elért eredményeket!

A dolgozatnak tartalmaznia kell:

- a feladat leírását,
- az estelegesen fellelhető, tervezést segítő eszközök rövid bemutatását,
- a feladatok és a köztük lévő kapcsolatok modelljének leírását,
- az tervet elemző algoritmusok leírását,
- a mintául szolgáló algoritmusok testre szabásának okait,
- a fejlesztést (általában) segítő technológiák bemutatását,
- az implementáció tervét,
- a tesztelési körülményeket, eredményeket,
- a programot és a működést bemutató prezentációt CD mellékleten.

Ph.

.....
Dr. Vámossy Zoltán
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022.05.15

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: LEX TAMÁS JÓZSEF

Neptun kód: [redacted]

Tagozat: NIK – S2F

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:
Virtuális Petri-ciklus

Szakdolgozat / Diplomamunka[2] címe angolul:
Virtual Petri-dish

Intézményi konzulens: Dr. habil. Szénási Sándor

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.03.05.	Téma pontosítása	[signature]
2.	2021.04.08	Irodalom bővítés áttekintése	[signature]
3.	2021.04.14.	Eredmények áttekintése	[signature]
4.	2021.05.06.	Rendszerterv véglegesítése	[signature]

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021.05.14.

[signature]
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:

Lex Tamás József ..... [redacted] ..... NIK-SZF .....

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Viruális Petri-csésze .....

Szakdolgozat / Diplomamunka[2] címe angolul:

Virtual Petridish.....

Intézményi konzulens: Külső konzulens:

Dr. habil Szénási Sándor ....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	**2021.09.23**	**Fejlesztési irány megbeszélése**	Szénási Sándor
2.	**2021.10.06**	**Implementáció bemutatása**	Szénási Sándor
3.	**2021.11.11**	**Források áttekintése, ábrák formázása**	Szénási Sándor
4.	**2021.11.23**	**Utolsó áttekintés**	Szénási Sándor

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 20.22.05.10.....

Szénási Sándor
.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

Tartalomjegyzék

1. Bevezetés

Életünk legapróbb építő elemei a sejtek. Ezek a kis építő elemek szabják meg a hétköznapjainkat, belőlük épül fel az egész élővilág, ezért megérteni azt, hogy milyen folyamatok játszódnak le bennük nagyon fontos. Működésük megértése nehéz feladat, ugyanis ahhoz, hogy vizsgálni tudjuk őket bonyolult eszközökre és komoly szaktudásra van szükség.

A számítógépek megjelenésével lehetőségünk nyílt arra, hogy ezeket a vizsgálatokat segítsük, vagy akár meg is valósítsuk egyet rajtuk. A számítógépen végrehajtott szimulációk haszna az, hogy bárhol meglehet őket tekinteni, nincs szükség hozzá speciális felszerelésre, labor környezetre. Bár az így elkészített kísérletek eredménye nem mindig pontos, de segíthet megérteni a belső folyamatokat, betekintést adhat egy olyan közönségnek, akik nem rendelkeznek a valós kísérleti eszközökkel, vagy akár egy közelítő eredményt adhatnak, amelyet használva a való világban is egyszerűbben megvalósíthatjuk a kísérleteket.

A sejtek szimulálása egy nagyon bonyolult és összetett folyamat, eddig még senkinek sem sikerült egy teljes sejt egy az egyben megvalósítása számítógép segítségével. A sejt működéseket vissza kell vezetni olyan formára, amelyen keresztül az ábrázolható az informatika eszközein, emiatt sok adat elveszhet, amely a valós környezetben jelen van. Viszont arra képesek vagyunk már, hogy egy-egy sejtfolyamatot elkülönítsünk, így lemásolva, hogy a sejtek hogyan működnek.

Dolgozatom célja, hogy bemutassam ezeket a lehetőségeket és létrehozzam a saját implementációmat a sejtek megvalósítására. Szeretném, ha modellem olyan sebességgel működne, hogy a sejtfolyamatok valós időnél gyorsabban menjenek végbe, lerövidítve a kísérletek idejét. Ezentúl szeretném még ezeket a kísérleti eredményeket megjeleníteni is, segítve azt, hogy megértsük hogyan is működik ez a kis mikrovilág. A létrehozott alkalmazásomban bemutatok egy példatárat, amely segítségével igazolom, hogy valós kísérletek használatakor is segítség lehet, ha számítógépen megvalósítjuk azokat.

2. Probléma analízise

2.1. Biológiai háttér

A probléma megértéséhez és a megoldás feltáráshoz szükség van a sejt pontos definíciójára. Dolgozatom első fejezetében csupán a sejt fogalmára, felépítésére s funkcióira kívánok kitérni, mely alapja a felvetett probléma megértésének, továbbá a megoldás feltárásának. Fontos megjegyeznem, hogy ezen szakdolgozatnak nem célja a sejtek témakörének teljes prezentálása - hiszen a sejtek tanulmányozásával egy külön tudományág a citológia [1] foglalkozik - ezért csak egy rövid áttekintőt kívánok szemléltetni.

2.1.1. Sejt felépítése

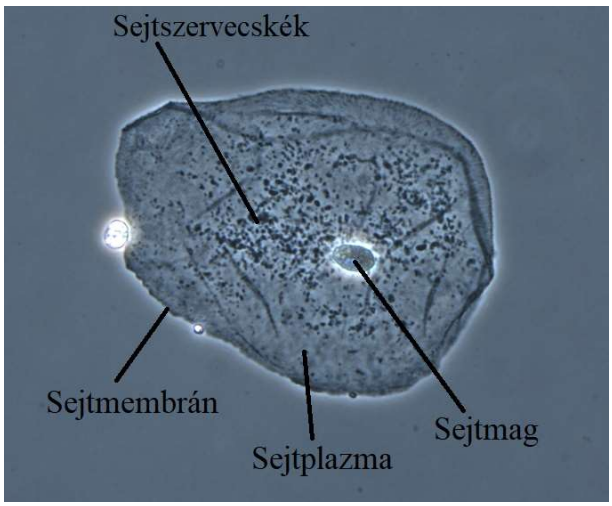
Minden lény, aki a földön él sejtekből épül fel: a baktériumoktól kezdve, növényeken át, egészen az emberig apró építőelemekből áll össze. Ezen okból kifolyólag a sejtek nagyon eltérő felépítéssel, alakkal jöhetnek létre. Egyesek egy sejtként alkotnak élő organizmust, míg mások több sejt kapcsolatából létrejött hálózatok segítségével alakítanak ki komplex működési formákat.

Általánosan elmondható, hogy a sejteket felépítésük szerint két sejttípusba sorolhatjuk: a kizárólag egysejtű életmódot folytató **Prokarióta** sejtek kevésbé komplexek, kevesebb DNS-t tartalmaznak, valamint nincs elhatárolt sejtmagjuk. Legrelevánsabb példaként a baktériumot tudnám kiemelni.

Ezzel szemben az **Eukarióta** sejtek egy és többsejtű életvitelre is képesek, örökítő anyagukat a külön sejthártya választja el a sejtplazmától, illetve több sejtszervecske különült el a sejtplazmán belül. Ide sorolhatóak a minket is felépítő sejtek összessége.

A sejtet egy külső fal, egy úgy nevezett **sejtmembrán** veszi körül. Ez különíti el a sejtet a külvilágtól, tartja egyben a sejtet és szabályozza az anyagáramlást a sejt és a külvilág között.

A membránon belül találhatóak a **sejtszervecskék** és a **sejtmag**. A sejtmag tartalmazza az örökítő anyagot, a DNS-t, mely tekinthető a sejt tervrajzának, ez alapján képes újra alkotni önmagát. A sejtszervecskék megtestesítői sejt szerveinek, mindegyik különböző funkcióval rendelkezik és nélkülözhetetlen a sejt életében. A sejtet felépítő elemek között egy vízszerű anyag, a **sejtplazma** található.



1. ábra: Sejt felépítése

2.1.2. Sejt funkciói

Sejtciklus az a folyamat, amelyet a sejtnek végre kell hajtani annak érdekében, hogy osztódni tudjon. Különböző sejteknek eltérő hosszúságú ez a folyamat. Egy átlagos emberi sejt körülbelül minden 24 órában végig megy ezen a folyamaton, de az emlősök között találunk olyan élőlényt, aki a gyors anyagcseréjének köszönhetően 9-10 óra alatt végrehajtja ezt a ciklust.

A sejtciklus általánosan 4 fázisra bontható: a G1 fázisban a sejt növekszik, megduplázza szervecskéit és felkészül a további lépésekre. S fázisban másolatot készít a sejtmagon belül tárolt DNS láncról. G2 fázisban a sejt újra gyarapodik tömegben. A sejtciklust az M fázis zárja, amely az osztódás folyamatát foglalja magában. Osztódás után a sejtciklus kezdődik előről, de létezik olyan sejt, amely képes megszakítani ezt a ciklust és osztódás után a G0 fázisba lép, amikor a sejt ellátja funkcióját, ám nem gyarapodik és nem osztódik tovább.

A **sejtosztódás** folyamata során az anyasejtből két leánysejt keletkezik. A sejtosztódásnak két formája a meiosis és a mitosis. Ezen dolgozatban a mitosis folyamatával foglalkozom, melynek folyamatakor a leánysejtek kromoszómái megegyeznek az anyasejt kromoszómáival. Ha hiba lép fel a mitosis során, akkor lehetséges, hogy az egyik leánysejt több míg a másik kevesebb kromoszómával rendelkezik, így emiatt hibásan látja el funkcióját.

Sejthalálnak nevezzük azt a folyamatot, amikor a sejt befejezi életfunkcióit, ez történhet külső hatások miatt, vagy valamilyen belső vezérlés eredményeképp. Programozott sejthalál azaz Apoptózis során a sejt lebontja önmagát utasításra, ezzel segítvén például az elöregedett sejtek kiszűrését. Nekrózis a sejt valamilyen sérülés vagy mérgezés (esetleg oxigén hiány) hatására történő elpusztulását jelenti. Ez a folyamat abban különbözik, hogy nincs vezérelve, véletlen is bekövetkezhet, illetve, hogy a folyamat során a sejtből távozhat a tartalma a

környezetbe és így akár megsértheti a szomszédos sejteket is. Érdekes információ, hogy a rákos sejtek képesek meggátolni az apoptózis folyamatát, így úgymond sosem halnak meg.

Bár a sejtek nem arról híresek, hogy maratoni távokat megtegyenek, de valamilyen szintű **mozgásra** képesek. A mozgás általában véletlenszerű, a külső környezet hatásai befolyásolják. Például, ha az az anyag, amiben a sejt tartózkodik megmozdul a sejt a mozog vele együtt. Ettől eltekintve léteznek olyan sejtek, amelyek képesek irányított mozgásra. A sejtek mozgásának ezt a formáját Taxisnak nevezzük mely akkor következik be, ha a sejt egy külső hatásra reakció mozgást hajt végre. A taxisnak különböző formái vannak, mivel a sejtek képesek különböző környezeti hatásokra reagálni. Példaként a következőket szeretném felhozni:

- Kemotaxis: reakció valamilyen vegyi anyagra (vonzás vagy taszítás toxin esetén)
- Fototaxis: mozgás a fény irányába, fény intenzitásától függően
- Termotaxis: mozgás a hőmérsékleti gradiens irányába
- Aerotaxis: reakció a környezet oxigén koncentrációjára

2.2. Létező megoldások sejtek szimulálására

2.2.1. Physicell

Physicell [2] szakít az eddigi hagyományos modellen alapuló megoldásokkal, a megalkotói egy Ágens alapú modellt hoztak létre, így kikerülve a később említésre kerülő Cellular Potts modellben bemutatott hálórendszert. A sejteket külön funkciókkal írják le, a térfogat és sejtek egymáshoz való tapadását is külön egységbe szervezték. Eredetileg rákos megbetegedések kutatására hozták létre, ám az így kialakult rendszer nem csak a rákos sejtek szimulációjára képes, hanem komplex szervi szimulációkat is létre lehet vele hozni. Ez a modell realisztikusabb szimulációs eredményeket ad, viszont implementációja, bővítése nehezebb és hardveres igénye is sokkal nagyobb. A program nem biztosít a kutatónak egy felületet, ahol módosítani lehetne a bemeneti paramétereket, ehelyett minden egyes szimuláció esetén szükség van a programkód módosítására, funkciók felülírására. Hagyományos megjelenítéssel sem rendelkezik, konzolos applikáció formájában fut, viszont képes exportálni Mathlab, XML és kép fájl formátumot egyaránt.

2.2.2. CellSim3D

Ez [3] a program egy több egyetemet is összefogó koalíció útján jött létre. A modell inkább a sejtek fizikai tulajdonságaira fókuszál, úgy, mint az őket összetartó erő, és különböző sejtek közötti műveletek során felszabaduló erők. Ennek a megvalósításnak fontos előnye, hogy az egész implementáció GPU-ra lett programozva (CUDA segítségével) és hogy képes Blender exportra és ott megjeleníteni a végeredményt.

Sajnos mivel nem rendelkezem Nvidia videokártyával, ezt a megoldást nem sikerült működésre bírnom.

2.2.3. Cellular Potts modell

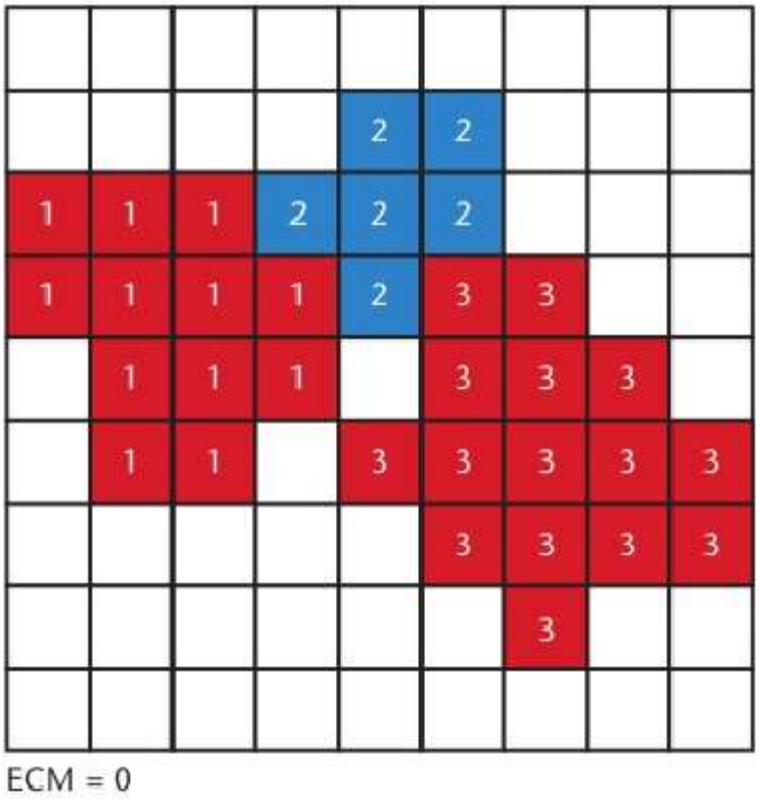
A Cellular Potts [4] modell sikeres használattal bír sejtek és más komplex biológiai rendszerek számítógépen való reprezentációjára. Számos alkalmazás használja ezt a modellt a sejtek szimulálására, így például CompuCell3D [5], valamint a Morpheus [6].

A modell a sejteket egy hálórendszer segítségével ábrázolja, ahol minden egyes mező egy számértékkel van ellátva. Az azonos számmal rendelkező mezők ugyanarra a sejtre utalnak. A szimuláció minimum energia elvére épül, azaz arra törekszik, hogy az energia mindig a legkisebb állapotban jelenjen meg a rendszerben.

Egy úgy nevezett Monte-Carlo *algoritmus* segítségével kiválaszt egy véletlenszerű mezőt a hálóból (i), majd ennek a mezőnek véletlenszerűen kiválasztja egy szomszédos mezőjét (j), ahová megpróbálja átmásolni a tartalmát. Abban az esetben, ha az új rendszer energia tartalma kisebb, akkor végbe viszi ezt a változtatást, ha viszont nagyobb vagy egyenlő, akkor egy valószínűségi számítás dönti el, hogy megtörténik-e a másolás vagy sem.

Az energia kiszámítására az algoritmus Hamilton-operátortorokat használ. Az operátor figyelembe veszi a sejt tapadási együtthatóját, tömegét és a sejtek közötti vonzási erőt.

Ezen tényező módosításával, új paraméterek hozzáadásával módosítható az sejtek viselkedése, így például elérhető a kemotaxis és a sejtkommunikáció folyamata is.



2. ábra: Cellular Potts modell mátrix reprezentációja

2.2.4. Chaste: Cancer, Heart and Soft Tissue Environment

A Chaste [7] nevéből is adódóan egy rákos megbetegedéseket kutató szoftvernek fejlődött. Opensource projekt, tehát bárki részt vehet a fejlesztésében. A program tartalmaz többfajta sejt megvalósítást, köztük a Cellular Potts és a Vertex modellt. Lehetőséget nyújt szív és tüdő betegségek vizsgálatára, ugyanis képes ezen szervek belső felépítését és működését sejt szinten szimulálni. Lehetőség van a tüdő oxigén ellátásának, anyag áramlásának szimulációjára. A szoftver úgy lett tervezve, hogy elosztott környezetben is optimálisan futtatható legyen. Nem rendelkezik felhasználó interfésszel, emiatt használata nehéz, ezenkívül nem egy általános megoldás a sejtek szimulációjára, hanem inkább a rákos megbetegedések vizsgálatára és a sejtek oxigén ellátásának vizsgálatára jött létre.

Megállapítások

A szakmai irodalom kutatása után azt állapítottam meg, hogy sejtek szimulációjára kétféle módszert használnak. Az első az Ágens típusú szimulációk, ahol a sejteket különböző részecskeként jelenítik meg. Ebben a megvalósításban a biológiai és kémiai folyamatokat pontosan tudni kell, megértésük és továbbfejlesztésük sokkal nehezebb, megvalósításnak az előnye, hogy pontosabb eredményeket kaphatunk vissza egy adott kísérletről.

A másik fajta megvalósítás valamilyen matematika módszert használ fel a sejtek szimulálására, ezek a módszerek azért lettek elkészítve, hogy segítsék a biológiai folyamatok megértését, számítógépes implementálásuk egyszerűbb a korábban említett társáénál.

3. Tervezés

A szakdolgozatom tervezése során először egy Ágens alapú implementáció elkészítésével próbálkoztam, ám az első prototípusok létrehozásakor sok nehézségbe ütköztem. Ezek a rendszerek nagyon közel állnak a valósághoz, ám létrehozásuk sok tudást és időt igényel. Legfőképp a biológiai tudással volt problémám, nagyon sok belső sejtfolyamatot nem tudtam értelmezni, emiatt a fejlesztés sokat állt és nem haladtam vele. Ezen nehézségek leküzdése érdekében kezdtem el kutatni újabb lehetőségek után, és ekkor találtam rá a fent is megemlített **Cellular Potts** modellre. Választásom végül ennek a modellnek a megvalósítására esett.

3.1. A kutatás célja

Szakdolgozatom célja egy olyan alkalmazás tervezése, megalkotása és tesztelése, amely képes a sejtek osztódásának és növekedésének, illetve rendeződésüknek a szimulációjára. A sejteket a szimuláció előtt paraméterezni lehet, meg lehet adni, hogy milyen funkciókra képes majd ezek után a szimuláció során képes valós időben megjeleníteni.

Az alkalmazásomnak rendelkeznie kell egy felhasználói interfésszel, amely lehetőséget ad a felhasználónak arra, hogy különböző beviteli mezők segítségével felparaméterezzen és elindítson egy szimulációt. A felhasználói interfésznek meg kell jelenítenie a szimulációt futás közben is és statisztikai eredményeket kell szolgáltatnia a felhasználó felé.

Az alkalmazásom fontos célja továbbá, hogy a szimuláció sebessége gyorsabb legyen, mint a valós kísérletek végbemenetele, így szükség van a modell párhuzamos megvalósítására.

A szakdolgozat teljesíthetőségének érdekében az implementálandó sejtfolyamatokat le korlátoztam a sejtek növekedésére, osztódására és rendeződésére egy adott rendszerben.

3.2. Szimuláció lépései

A Cellular Potts modell egyik nagy előnye, hogy szimulációs lépései jól dokumentáltak. Az első lépés, hogy létre kell hozni egy hálót, és minden egyes elemének adni kell egy azonosító számot. A nullás azonosítót a környezetnek szokták adni, a többi mező, ami nullától eltérő értékkel rendelkezik részt vesz a szimulációban. Az azonos számmal rendelkező mezők ugyan azt a sejttípust jelölik.

Az alap modell az adhéziós erőket és a sejtek belső térfogat változását veszi figyelembe. Az algoritmus figyeli az ebből eredő energiaváltozásokat és próbálja ezt az energiát minimalizálni.

A szimulációkat alapvetően két csoportba sorolhatjuk a determinisztikus és a sztochasztikus. A determinisztikus szimulációk során minden esetben ki tudjuk számítani egy adott bemenet hatására keletkező kimenetet, így minden ismétlésre ugyan azt az eredményt adják. A

sztochasztikus szimulációk során valamilyen véletlenszerűség bekerül a rendszerbe. Ezt a véletlenszerűséget jellemezhetjük egy valószínűségi változóval.

Mivel a sejtek életmodellje erősen sztochasztikus folyamat, emiatt a szimulációmban is szükséges a véletlenszerűség bevitele. Ennek a problémának a megoldására a modellem implementálni fogja a Monte-Carlo módszert.

3.2.1. Monte-Carlo lépés

A Monte-Carlo algoritmus egy ismételt véletlen mintavételezésen alapuló megoldás. A folyamat során véletlenszerűen kiválasztunk egy mezőt majd ennek a mezőnek egy véletlenszerű szomszédját. Megpróbáljuk a mezőt átmásolni ebbe a szomszédos mezőbe. Kiszámítjuk az energia differenciát az így létrejött új és a régi rendszer között. Ha ez az energia különbség kisebb, mint nulla, akkor mindenképp alkalmazzuk ezt az új rendszert. Ha viszont 0 vagy annál nagyobb, akkor nem visszük végbe a változást. Az így kialakult rendszer egy idő után egy energia minimumba rendezi magát, viszont annak elkerülésére, hogy egy lokális minimumba ragadjunk alkalmazunk egy valószínűségi számítást. A számítás paraméterezhető egy T változóval, amely a rendszer **Boltzmann-állandója** és tekinthető a szimuláció hőmérsékletének is. Ez a megvalósítás nagyban hasonlít a szimulált lehűlés optimalizációs módszerre, amelyet Haladó Algoritmusok tárgy keretében tanultunk.

Egy adott rendszer energiatartalmának kiszámítására a Hamilton-operátorok összegét használjuk. Az alap modellben szereplő egyenlet figyelembe veszi az adhéziós erőt az azonos sejttípusok között, az adhéziós erőt a különböző sejttípusok között és végül a sejtnek a térfogatát és annak változásait.

Ez az egyenlet bővíthető új paraméterekkel, amelyek így képesek újabb és újabb viselkedés leírására. Az alkalmazásom ezeket a paramétereket módosítókként fogja ábrázolni. Egy sejtnek lehet adni módosítót, amely így különböző viselkedés formákra bírható.

3.2.2. Implementálandó bemeneti paraméterek és Hamilton-operátorok

Ahhoz, hogy a modell megfelelően működni tudjon különböző módosítókat kell alkalmaznom, amelyek az energia számításkor befolyásolják az eredményt. Ezek a módosítók minden egyes másolás előtt végig futnak a rendszeren kiszámolva az energia változást. A megoldásom az alábbi módosítókat fogja implementálni.

$$H = \sum_{i,j \text{ neighbors}} J\left(\tau(\sigma_i), \tau(\sigma_j)\right)\left(1 - \delta(\sigma_i, \sigma_j)\right) + \lambda \sum_{\sigma_i} \left(v(\sigma_i) - V(\sigma_i)\right)^2,$$

3. ábra: Az alap Cellular Potts modell képlete. A J paraméter befolyásolja a sejtek közötti adhéziós erőt a V paraméteres rész pedig a tömeg változásért felelős

Adhéziós módosító a legelső, amely megjelenik a Potts modellben. Egymagában, bármiféle más módosító nélkül, véletlenszerűen kiosztott mező értékekkel kiadja az Ising modellt. Az Ising modell a legegyszerűbb módja annak, hogy a termodinamika törvényeit bemutassuk, ugyanis egy állapotváltozást mutat be, ahol mindig próbálja a rendszert a legkisebb energia állapotba helyezni. Az adhéziós módosítónak N darab paramétere van sejtenként megadva, ahol N a sejtek számát jelenti. A legelső beviteli mező mindig az adott sejt környezethez viszonyuló adhéziós erejét adja meg, a következő N db pedig az adott i-k sejtre vonatkozó adhéziós erőt szabja meg. A modellben szükség van megadni a saját sejtfajtájához szóló adhéziós erőt is, de ez általában egy „nagyon nagy" szám, így elérve, ahogy az azonos sejtek egybe maradjanak. Ezek a bemeneti paraméterek futásidőben változtathatóak, ha például szeretnék az Immunsejtek ciklusát szimulálni. Az immunsejtek egy vizsgálat után el tudják dönteni egy adott sejtről, hogy idegen vagy nem. Ha ennek a vizsgálatnak az eredménye az, hogy a sejt idegen, onnantól kezdve jobban fognak „tapadni" ahhoz az adott sejthez, hogy meggátolják tovább terjedését.

Tömeg módosító az adhéziós módosítóval együtt adja ki az alap Cellular Potts modellt. A tömeg módosítónak két bemeneti paramétere van sejtenként: az első a V paraméter megadja a sejt maximális „tömegét", melyet elfoglalt cellák összege adja meg. A második paraméter a λV pedig megadja, hogy egy adott szimulációs ciklus alatt mekkora növekedésre képes egy adott sejttípus. Ez a paraméter lehet konstans vagy a sejt környezetétől függően változhat, alkalmazkodva a tápanyag változásokhoz. Így elérhető egy olyan szimuláció, ami optimális környezetben történik, a sejt folyamatos növekedésnek van kitéve, vagy pedig egy olyan szimuláció, amely közelebb van a valósághoz és a sejtek a környezetben lévő tápanyagok alapján képesek növekedni.

Az **Osztódási módosító** befolyásolja a sejt osztódási sebességét. Ha a sejt elér egy bizonyos tömeghatárt, akkor két új leánysejt jön létre belőle, amely megörökli a szülősejt eredeti paramétereit. Ha ez a paraméter 0, akkor a sejt sosem fog osztódni.

„Act" módosító befolyásolja azt, hogy egy sejt mennyire aktív. Minél nagyobb ennek az értéke, a sejt annál inkább mozogni fog az adott területen.

T paraméter a szimuláció hőmérsékletét adja meg. Minél magasabb ez az érték, annál aktívabb lesz a sejtek mozgása, mivel a negatív energiaváltozással járó folyamatok nagyobb eséllyel mennek végbe.

3.3. Rendszer terv

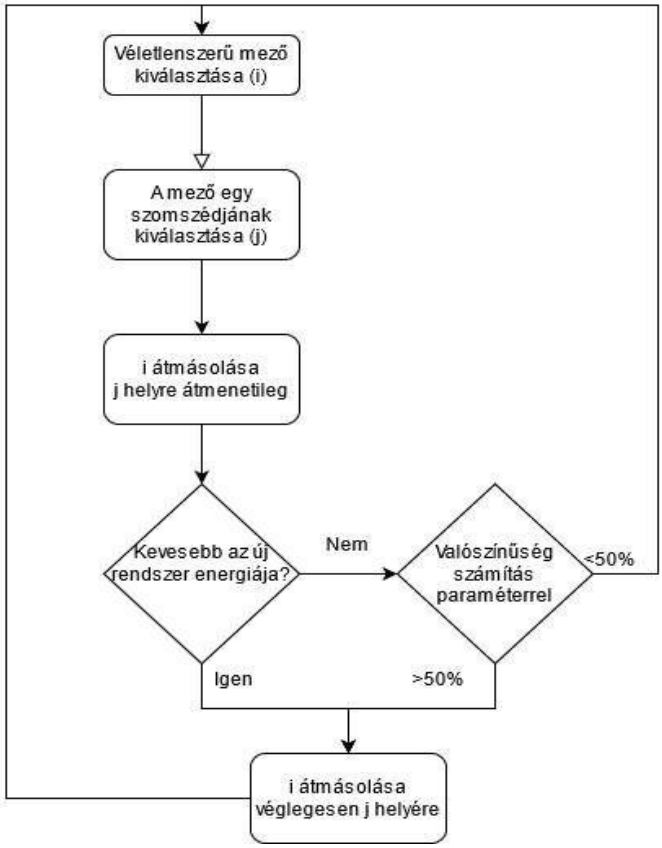
Az én projektemben az értékét álláspontom szerint a modell kiválasztása, megvalósítása és párhuzamosítása adja. A Cellular Potts modell nagyban hasonlít a Haladó Algoritmusok tárgy keretében tanult Szimulált lehűlés technikára, ezért ebből indultam ki a jelen ábra szerint.

3.1. Algoritmus SimulatedAnnealing

```
1:  függvény SIMULATEDANNEALING(𝕊, d_s, f, ϵ, k_B, TEMPERATURE,
                STOPCONDITION)
2:      p ←(rnd) 𝕊
3:      p_opt ← p
4:      t ← 0
5:      ciklus amíg ¬STOPCONDITION()
6:          t ← t + 1
7:          q ←(rnd) {x ∈ 𝕊 | d_s(x, p) = ϵ}
8:          ΔE ← f(q) − f(p)
9:          ha ΔE ≤ 0 akkor
10:             p ← q
11:             ha f(p) < f(p_opt) akkor
12:                 p_opt ← p
13:             elágazás vége
14:         különben
15:             T ← TEMPERATURE(t)
16:             P ← e^(−ΔE/(k_B T))
17:             ha RND_u(0, 1) < P akkor
18:                 p ← q
19:             elágazás vége
20:         elágazás vége
21:     ciklus vége
22:     vissza gpm(p_opt)
23: függvény vége
```

4. ábra: Szimulált lehűlés algoritmusa

A szimuláció hőmérsékletét állandóra állítom és a szimuláció magjában fognak végbe menni a Hamilton-operátor által történő energiadifferencia számítások az optimalizációs folyamatok helyett.



5. ábra: Folyamatábra

A fent megtekinthető folyamatábra tartalmazza a szimuláció lépéseit egymás után.

Az elkészült szoftver 3 részből fog felépülni. Szeretném az áttekinthetőség miatt elkülöníteni a megjelenítést, magát a modell megvalósítását, illetve a videókártyával történő kommunikációt és optimalizációt.

A megjelenítést oly módon fogom elkülöníteni a szimulációtól, hogy a kettő sebessége ne befolyásolja egymást. Így elérhető az, hogy bár a magas számítási igényekkel bíró folyamatok a háttérben lassabban fussanak, maga a megjelenés folyamatos és azonnali legyen. Ezt oly módon fogom elérni, hogy a szimuláció és a megjelenítés két külön konkurens szálon fog futni. A kirajzoláshoz használandó adatokat úgy fogom kiszolgálni a szimulációs szálból, hogy ott versenyhelyzet ne alakulhasson ki. A megjelenítést ezentúl ki fogom egészíteni egy videókártyára történő optimalizációval is, a kapott adathalmaz képpé alakítását azon fogom végezni. A megvalósítandó módszer segítségével a képek létrehozása azonnalinak fog hatni a felhasználó szemszögéből, ugyanis az erre készített hardver képes az adatokon párhuzamos módon műveleteket végezni, a CPU hagyományos lineáris iterációjával szemben.

A szimuláló szál sebességét oly módon fogom növelni, hogy az egyszerre nem egy, hanem több Monte-Carlo lépést hajtok végre. Ennek a megvalósítása úgy fog történni, hogy létrehozok egy szál gyűjteményt, amelyet a szimulációs során folyamatosan feladatokkal fogok ellátni. Így egy időben több Monte-Carlo folyamat is megtörténhet. Itt fontos feladat lesz a versenyhelyzet detektálása és kivédése, ennek a megoldására alkalmazni fogom tanulmányaim során tanult eszközöket.

4. Eszközök kiválasztása

4.1. Programozási nyelv

A megvalósításom készítésekor sok tényezőt kellett figyelembe venni, de a fő szempont a gyorsaság a nagy számításigényű folyamatok során. Választásom végül a C++ programozási nyelvre [8] esett, mivel támogatja a magas nyelvi funkciókat, illetve a GPU programozást, de képes a memória alacsony szintű hozzáférésére is. Az ebből eredendő teljesítmény növekedések nélkülözhetetlenek a feladatom megvalósításakor.

4.2. GPU programozás

A számítógépünkben lévő videokártya nem csak a grafikai megjelenítésre képes, hanem párhuzamos feldolgozói segítségével bonyolult matematikai és programozási műveletek végrehajtására is. Ezt a tulajdonságot kihasználva jelentős gyorsulás érhető el bizonyos feladatok megvalósítása során. A GPU-ra való programozáshoz két eszköz található a piacon a CUDA és az OpenCL [9] . A CUDA csak az Nvidia videokártyák által van támogatva, az OpenCL ezzel ellentétben egy nyílt forrású GPU programozási eszköz. A piacon lévő legtöbb gyártó támogatja valamilyen verzióját, ebből eredendően több eszköz elérésére is képes a CUDA-val szemben. Hátránya, hogy minden gyártó saját implementációt készít belőle és több gyártó is (például: Intel) elzárja saját megvalósítását. Ezentúl hátránya még, hogy a keretrendszere nincs olyan kiforrott formában, mint a CUDA, emiatt a programozónak, aki használni akarja több munkát kell végeznie.

Az én számítógépem AMD és Intel videokártyákkal van ellátva, ebből az okból kifolyólag én az OpenCL használata mellett döntöttem a hátrányai ellenére is.

4.3. Megjelenítés

A megjelenítéshez és felhasználói interfész elkészítéséhez egy olyan opciót szeretnék kiválasztani, ami nem veszi el az erőforrást magától a szimulációtól. Több C++ könyvtár megvizsgálása után a választásom a Dear Imgui [10] nevezetű csomagra esett.

A Dear Imgui egy olyan fejlesztői könyvtár, amely azt célozta meg, hogy minél több függőséget kihagyjon, könnyű legyen őt használni, képes legyen több grafikus motor használatára is és csak egy fájt kelljen egy projektbe beimportálni.

A választásom megerősítéséhez korábbi tapasztalataim is hozzájárultak. Az applikációmat Windows-os környezetben fejlesztem, emiatt a grafikus motorként a DirectX [11] könyvtár eszközkészletét veszem még használatba.

4.4. Verzió kezelés

A verzió kezelés fontos része egy modern alkalmazásnak. Az új funkciók kezelése és legfőképpen a forráskód mentésének céljából sok alkalmazás ajánl szolgáltatást. Az én választásom a GIT-re [12] esett korábbi tapasztalatok és a branching modell használata miatt. Git 2005-ben lett készítve Linus Torvalds által a Linux kernel fejlesztésének támogatása céljából. Azóta az ipar fontos része és sok neves cég is használja az alkalmazásának verzió kezelésére.

A git mellett használni fogom a Github szolgáltatásait, mely webes szolgáltatásokat nyújt a git használatára. Segítségével egy internetes tárhelyen tárolhatom a kódomat és oszthatom meg másokkal.

5. Elkészült szoftver bemutatása

5.1 Párhuzamos Cellular Potts modell

A megvalósított modellem alapja egy 2 dimenziós mátrix, amelyet a Grid osztály valósít meg. Magát az értékeket egy 1 dimenziós tömbben tárolom a jobb teljesítmény elérésének érdekében, emiatt viszont szükségem volt arra, hogy egy 2 dimenziós koordináta rendszerből tudjak konvertálni 1 dimenziós indexé. Ebben az osztályban kezelem még azt az esetet is, ha egy index szomszéd értékeire van szükség. A Grid értékeit módosítja, illetve optimalizálja a modell.

A gyorsabb működés érdekében eltárolom azokat a mátrix értékeket, amelyek a sejt „szélén" állnak. Erre végzek egy számítást, amely során végig megyek az összes értéken és ha a saját típusánál eltérő típus található az adott index közvetlen közelében, akkor ez egy „sejt szélnek" tekinthető. Ezeknek az értékeknek a helyét eltárolom egy külön osztályba, amelyet DiceSet néven neveztem el, mivel ő felel a véletlenszerűség kialakításáért. A gyűjteményhez hozzá lehet adni egy értéket, törölni lehet egy értéket, illetve végre lehet hajtani egy mintavételezést. A mintavételezés során visszaad egy véletlenszerű értéket a gyűjteményből. Korábbi eredményeim során azt vettem észre, hogy a C++ rand() függvényét használva a véletlen szám kialakítására visszatérő sejt alakzatok keletkeztek, a sejt „formája" ismétlődött. Ez azt jelentette számomra, hogy a véletlenszerűség eloszlása nem volt megfelelő, emiatt újabb véletlenszám generálás után kellett kutatnom. A végső választásom a Mersenne Twister [13] nevezetű algoritmus megvalósítására esett. Ez a pszeudo-véletlen generátor sokkal jobb eloszlással rendelkezik, mint az alap C++ által megvalósított verzió, emiatt a szimulációk eredménye sokkal realisztikusabb lett, minden egyes futás során más és más sejt formák jönnek létre, nincs visszatérő alakzat.

A modell fő lépésekor ezekből a sejt szélekből hajtok végre egy mintavételezést majd módosítom az adott szél értékét a Hamilton operátorok eredményének függvényében. A Monte-Carlo lépés végig megy az összes szél értéken véletlenszerűen kiválasztva, majd, ha végzett, akkor frissíti a szél értékek gyűjteményét az újonnan kialakult állapotra. A Monte-Carlo lépés legvégeként végig megyek az összes kialakult sejten és ha osztódás be van állítva, akkor azt is itt hajtom végre.

Az elkészült Hamilton operátorok listája a következő:

- Adhéziós módosító
- Tömeg módosító
- Aktivitás módosító
- Perem módosító (Peremiter)

Minden egyes módosítóhoz tartozik egy paraméter, amely segítségével módosítható a működése. Az összes módosító megvalósít egy ősosztályban definiált DeltaH függvényt, amely bemenetként fogad egy forrás indexet és egy cél indexet, amely kettő közt megy végbe a számítás. A számítás eredményeitől függ a szimuláció további lépése. Ha az eredmény pozitív, az azt jelenti, hogy a módosítás energia szempontjából nem előnyös, mivel több energiára van szükség a végrehajtásához, mint amennyi most jelenleg a rendszerben van. Ha ez eredmény negatív, akkor viszont az azt jelenti, hogy energiát használunk fel, a rendszer összességének az energiája csökkenni fog. Mi erre az energia csökkenésre törekszünk, ezért a rendszert az új állapotba hozzuk, átmásoljuk a forrás index értékét az cél index helyére.

5.1.1 Adhéziós paraméterek

Az Adhéziós módosító paraméterei befolyásolják azt, hogy az adott típusok egymás közt milyen adhéziós viszonyban vannak. A paraméterben meg kell adni a környezet és egy adott sejttípus, illetve az sejttípusok egymás közti erőviszonyát. Azt tudjuk, hogy a 0 érték a környezetet jelenti. Tegyük fel, hogy az 1-es típusú sejt paramétereit akarom beállítani. Definiálnom kell, hogy a 0->1 illetve a 1->1 között mekkora az adhéziós erő. Minél nagyobb pozitív számot írok be, annál jobban fog egy adott irányba vonzódni a sejt. Ha negatív értéket adok meg, akkor a sejt távolságot fog az adott érték felől. Ha a környezethez fűzött értékét módosítjuk, akkor megadhatjuk, hogy egy sejt mennyire legyen folyékony vagy tömör.

5.1.2 Tömeg paraméterek

Egy sejt tömegét az adja meg, hogy hány cellányi helyet foglal el. A sejt tömegeket minden egyes Monte-Carlo lépés után kiszámítom és eltárolom. A módosító paramétereinek meglehet adni azt, hogy az adott sejt mekkorára nőhet (hány cellát foglalhat el), illetve, hogy egy lépés alatt hány cellányi értéket növekedhet. Ez az érték lehet negatív is, ekkor a sejt tömege csökkeni fog. A sejt tömege azért fontos, mivel e paraméter szerint függ a szaporodás sebessége is.

5.1.3 Aktivitás paraméterek

Aktivitás módosító hozzáadásakor meg kell adni, hogy egy adott sejt melyik értékek irányába szeretne vagy nem szeretne mozogni. Példaként van két sejttípusunk az 1-es és az 2-es. Ha megadjuk, hogy az 1-es sejttípus valamilyen pozitív értékben „vonzódik” a 2-es felé, illetve a 2-es valamilyen negatív értékkel „taszítódik”, akkor elérhetjük, hogy a sejtek kergessék egymást a szimulációs térben.

A további módosítóknak nincsenek bemeneti paramétereik, a peremmódosító segítségével azt mondom meg, hogy a sejt ne próbálja meg a szimulációs teret elhagyni.

A szimuláció sebessége függ attól, hogy magán a szél értékeken milyen gyorsan megyünk végbe, emiatt magát a szél érték iterációt egy párhuzamos megvalósítással hoztam létre. Az értékeken több konkurens szál megy végig, a széleket külön-külön vizsgálva. Mivel egy sejt a

való életben sem egy irányba indul el, emiatt a konkurens megvalósítás még pontosabb biológiai modellt is ad vissza, a sejt képes több irányba is elindulni egyidejűleg. A DiceSet mintavételezési függvénye meggátolja annak a lehetőségét, hogy két szál ugyan azt az értéket akarja módosítani, kizárva a versenyhelyzeteket.

5.2 Megjelenítés és kommunikáció a videókártyával

Megjelenítés, illetve ennek a sebessége nagyon fontos volt számomra modell kialakításakor. Szerettem volna egy olyan programot létrehozni, amelyben a sejtek változása a szemlélő láttára mennek végbe, nem kell várnia az eredményre.

A megjelenítéshez szükség van az előző fejezetben bemutatott rács kirajzolására a képernyőre. Az értékeket egy 1 dimenziós tömbben tároltam, amely tekinthető egy kép adat buffernek is. Ezt a buffert adom át a videókártyának és dolgozom fel párhuzamosan. A videókártya sokszoros párhuzamosítása miatt képes a pixel értékeken nagy sebeséggel végig iterálni és létrehozni a végső kép halmazt. Minden értéket megvizsgál és az alapján eldönti, hogy a végső képnek az adott pixele milyen színű legyen. A végeredménye ennek a számításnak egy kép, amelyet kirajzolok a képernyőre.

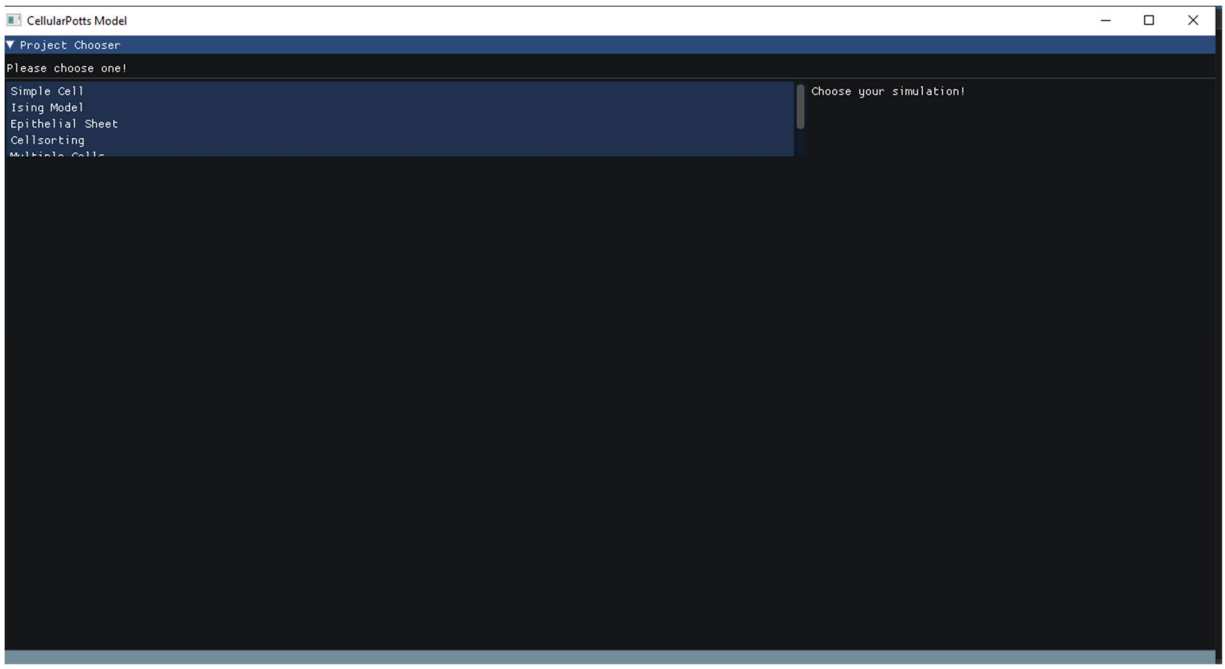
A kirajzolás emiatt nem függ attól, hogy a szimuláció éppen milyen állapotban van, mindig képes az adott pillanatot kirajzolni. A megjelenítés ezenkívül nem függ a szimuláció sebességétől sem, a szimulációs lépések más sebességgel futnak, mint maga a kirajzolás.

A videókártyával való kommunikációt OpenCL 2.0 [9] segítségével oldottam. A program ezen része feltérképezi, hogy milyen OpenCL kompatibilis eszközök találhatóak az adott környezetben, kiválasztja az elsőt és magát a számításokat ezentúl azon az eszközön végzi.

5.3 Felhasználói interfész bemutatása

A felhasználói interfészem kialakításakor törekedtem arra, hogy átlátható, könnyen kezelhető és reszponzív legyen. Fontos, hogy a felhasználó egy adott változtatása megjelenjen a felületen és ne kelljen minden opciót kutatnia.

Az alkalmazás elindításakor egy lista fogad minket a megvalósított kísérletekről. Itt kiválaszthatjuk, hogy melyik kísérletet szeretnénk végrehajtani.

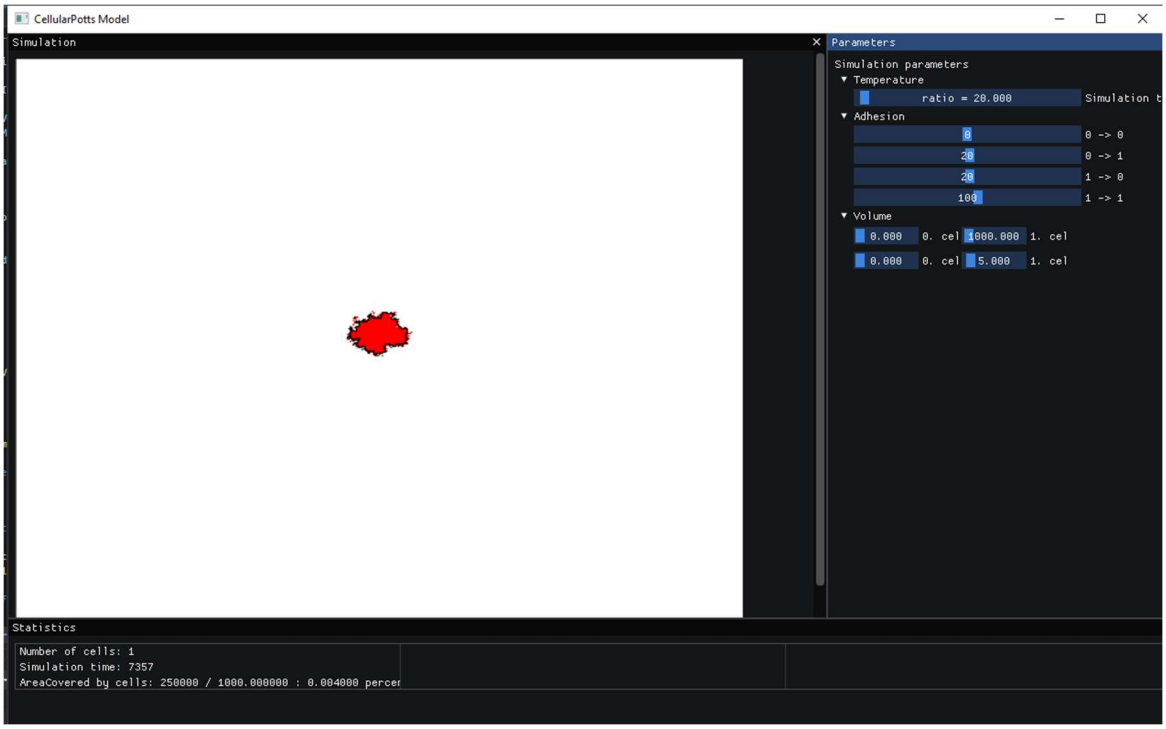


6. ábra: Program elindítása után ez a képernyő fogad. Itt található a szimulációk listája.

A kísérlet kiválasztása után az rögtön megjelenik a képernyőn. Szerettem volna azt, hogy maga a megvalósult szimuláció legyen a központban, mivel a programom előnye a legtöbb sejtszimulációval szemben, hogy valós időben történik.

A megjelenített szimuláció jobb oldalán találhatóak a szimuláció változtatható paraméterei. Egy lenyíló menüben minden bemeneti paraméter értékét tudjuk változtatni a futási idő közben. A szimuláció alatt pedig statisztikák jelennek meg az eltelt szimulációs időről, a képernyőn

látható sejtek számáról, illetve arról, hogy ezek az adott sejtek mekkora területet fednek le az egész szimulációs térből.



7. ábra: Ez a képernyő jelenik meg a szimuláció futása során.

6. Kísérletek végrehajtása

Ebben a fejezetben szeretnék bemutatni néhány kísérletet, amelyet végre lehet hajtani a programom segítségével. Az alkalmazás fő célja egy sejtnek a szimulálása, viszont a különböző bemeneti paraméterek változtatásával, illetve a sejtszám növelésével komplexebb kísérletek is végre lehet hajtani. Először szeretném bemutatni, hogy egy sejt esetén mi történik a bemeneti paraméterek változtatásának hatására, majd bemutatni azt, hogy a szaporodás milyen módon lett implementálva, illetve, hogy a növekedési paraméterek milyen hatással vannak egy-egy sejt szaporodására.

Végül szeretném bemutatni egy olyan kísérletet, amely több sejtnek az egyidejű szimulációjával történik, ezzel bizonyítva, hogy az elkészült modellem használható akár olyan kísérletek számítógéppel való szimulációjára, amelyeket amúgy csak hagyományos labor körülmények között tudnánk végrehajtani. Fontos, hogy az én általam kapott eredmények hasonlóak legyenek a laborban mérhető eredményekhez képest. Ennek ellenére szeretném megjegyezni, hogy az én modellem nem tökéletes, illetve, hogy a laborban kapott eredmények több külső tényezőtől is függenek, emiatt kaphatok eltérő, vagy csak közelítő eredményeket is. Az én szimulációmban a sejteknek sem tápanyagra, sem oxigénre nincs szüksége, hiszen egy „ideális” környezetet alakítok ki nekik. Példaként egy sebgyógyulás-t fogok végrehajtani, mivel egy emberi seb gyógyulásakor az optimális esetben a sejteknek minden szükséges tápanyag a rendelkezésükre áll, emiatt a környezet hasonlít az én programomban kialakított környezetre.

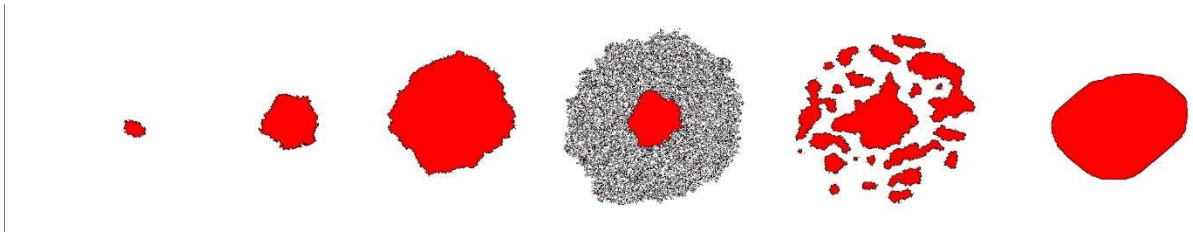
A szimulációkat egy laptopon fogom végrehajtani, mivel tudom, hogy több kutatólaborban hagyományos asztali számítógépek helyett jobban preferálják az hordozható eszközök használatát kényelmi szempontok miatt. Ily módon bizonyítani tudom azt is, hogy az általam készített szoftver használható kevésbé erős hardveri környezetben is. Az eszközöm processzora egy 8. generációs Intel alacsony fogyasztású laptop processzor az I5-8265U kódnévvel. Az eszközben nem található dedikált grafikus gyorsító, emiatt a processzorba integrált grafikus gyorsító segítségével fogom végrehajtani a szimulációkat. Az eszköz ezenkívül 8GB memóriával rendelkezik.

6.1 Egy sejt szimulálása

Elsőként szeretném bemutatni, hogy mi történik egy sejtnek a szimulálásának esetén. A szimulációs térben elhelyezek egy sejtet, és adhéziós, illetve tömeg módosítót be paraméterezem. A szimuláció futtatása után látható, hogy egy piros „folt” jelenik meg, amely szélein mozgást végez. Mivel ennek a sejtnek semmilyen olyan módosítót nem kapcsoltam be, amely bármiféle sejtfunkcióra késztetné, emiatt csak a véletlenszerű sejtmozgást végez.

Ha elkezdem módosítani a paramétereket, akkor elérhető valamiféle változás a sejtnek a megjelenésében. Ha a tömeg módosítót a sejtnél pozitív irányba módosítom, akkor látható,

hogy a sejt elkezd növekedni, ha pedig ezt a módosítót negatív irányba mozgatom, akkor elérhető, hogy a sejt mérete csökkenjen. Az adhéziós paraméterek változtatása esetén elérhető, hogy a sejt „szét essen”, mivel az őt összetartó erők megváltoznak, emiatt a sejt belseje szétesik. Ennek az ellentétje, ha a belső vonzást nagyobb értékre állítom, akkor a sejt jobban „össze áll”, vagyis a belsejében lévő részecskék közelebb lesznek, kevésbé laza kapcsolat alakul ki, illetve a véletlenszerű sejtmozgás is lelassul, akár meg is állhat. Ha a környezet adhéziós paramétereit állítom, akkor pedig ezek a változások a sejtnek a külsején mennek végbe, azt jelentve, hogy a sejt fala fog szétfolyni, illetve a sejt fala fog összetömörülni. A kettő között a különbség, hogy míg az első változás a sejt belsejéből történik, addig a második egy külső hatásra történik meg. A kísérlet megjelenítéséről készítettem képeket is, amelyeken látható, hogy a különböző bemeneti paraméterek milyen sejtformákat hoznak létre.



8. ábra: Különböző sejtformáció a paraméterek hatásaira

6.2 Sejtek szaporodásának szimulálása

Egy sejt szimulálásánál már csak több sejt szimulálása érdekesebb. Több sejt esetén megnőnek az erőforrás igények, komplexebb lesz az egész szimulálás folyamata.

Az életben egy sejtnek, ha minden feltétele megvan, akkor populációja folyamatosan növekedő tendenciát mutat. Minden sejt szaporodik és próbálja kitölteni az adott teret. A szimulációmban állítható, hogy a sejtek szaporodása mikor következzen be. Ez azért fontos mert különböző sejttípusok a növekedési fázis különböző szakaszaiban hajták végre osztódásukat. Több sejt estén megfigyelhető a szimulációban, hogy a sejtek egymást nem fedik át, hanem csoportokba rendeződnek. A szaporodás belülről kifelé történik, de a szimuláció megvalósítása miatt a belső sejtek és képesek további szaporodásra. A kísérlet bementi paramétereit úgy állítom be, hogy egy igen gyorsan szaporodó sejt életciklusát mutassam be. A szimulációs teret igen nagyra, 1000x1000 pixel méretűre állítom. Ez lehetővé teszi, hogy nagyobb számú sejtek szimulációját is bemutassam, mivel lesz tere a sejtekben szaporodni. Egyfajta sejt fog részt venni ebben a szimulációban és indulásképpen az egyedszáma is egy lesz. A sejtek egy szimulációs kör alatt 15 egységnyit képesek maximálisan növekedni, és a legnagyobb méret, amit elérhetnek az 1500 egység. A tér nagysága miatt ezekből a sejtekből

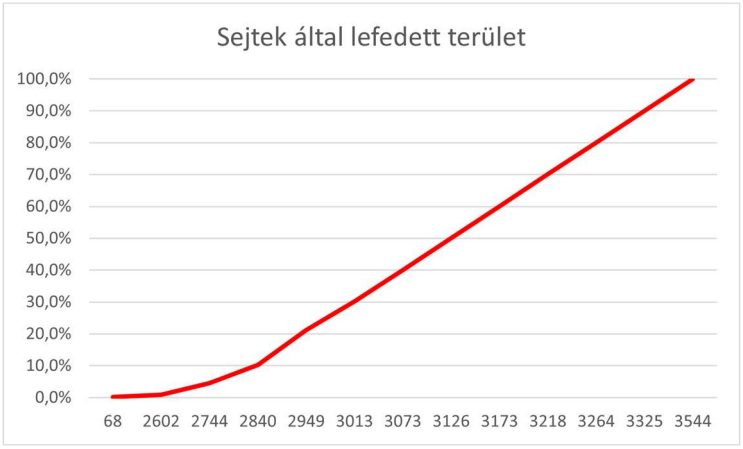
feltételezhető, hogy optimális esetben, ha az összes sejt a legnagyobb tömeget eléri, akkor 666 db lesz. Ez az az sejtszám, amely képes 100% mértékben kitölteni a teret. Fontos bemeneti paraméter még a szimuláció hőmérséklete, amelyet 20 „fokra” állítok, így a sejt aktivitás optimális értékeket érhet el. Nem lesz sem túl kiugró sem túl lassú. A sejtek közötti adhéziós erőket oly módon állítom be, hogy a különböző sejtek akarjanak egymás mellet lenni, így elérve, hogy szép rendeződések alakulhassanak ki. A sejt és a környezet közötti adhéziós erőt semlegesen állítom be, így a sejtek nem fognak sem szétfolyni sem összetömörülni. A paraméterekből látszódik, hogy a sejtnek tömeg és adhéziós módosítót állítok be.

A szaporodás a szimulációban oly módon állítható be, hogy megadható az a kritikus tömeg, amely felett a sejt szaporodást fog végrehajtani, illetve megadható egy valószínűség is ezen kívül, amely még jobban szűkíti a szaporodási esélyt. A valós életben is megfigyelhető valamilyen valószínűség egy sejt szaporodási folyamataira, e módon beállítható, hogy egy sejt milyen gyorsan duplikálja magát.

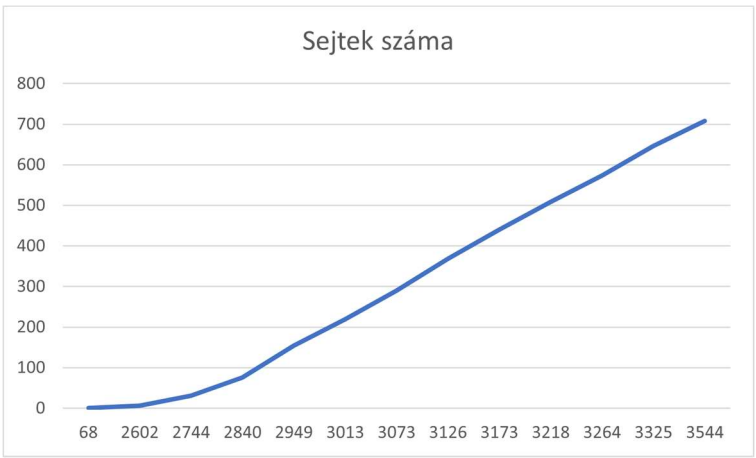
Én ebben a szimulációban a kritikus tömeget a sejt maximális tömegének a 95%-ára állítottam, illetve, ha a sejt eléri ezt a tömeget, akkor 10% esélye van a szaporodásra. Így elérhető, hogy sejtek ne minden szimulációs lépés végén szaporodjanak, ha elérték az adott tömeget, és jobban elkülöníthető, megfigyelhető a folyamat.

A szimuláció futtatása után végig nézhető a folyamat. Az kísérletet addig futtattam, amíg a sejtek el nem foglalták az egész területet. A szimuláció 19 percig tartott, ami idő alatt 3544 db szimulációs lépés futott le. A végső sejtek darabszáma 708 darab, amely különbözik a megjósolt 666 darabtól. Ez azért történhetett, mivel a sejtek már a maximális tömegük elérése előtt képesek szaporodni, viszont a terület nagysága véges. Az is megfigyelhető, hogy a lefedett terület nagysága, illetve a sejtek számának növekedése hasonló diagrammokat produkál. Ebből következtethető a sejtek száma kapcsolatban van a lefedett terület nagyságával.

Szeretném még azt is megjegyezni, hogy a kiindulás után sok idő eltelt mire az első sejt osztódott, mivel neki egyedül kevés volt az esélye, hogy szaporodjon. Az első nagy szaporodási robbanás után viszont a sejt számosságából kiindulva megnőtt az esély a sejtek szaporodására és onnantól megfigyelhető volt egy lineáris kapcsolat a sejtek száma és az eltelt idő között.

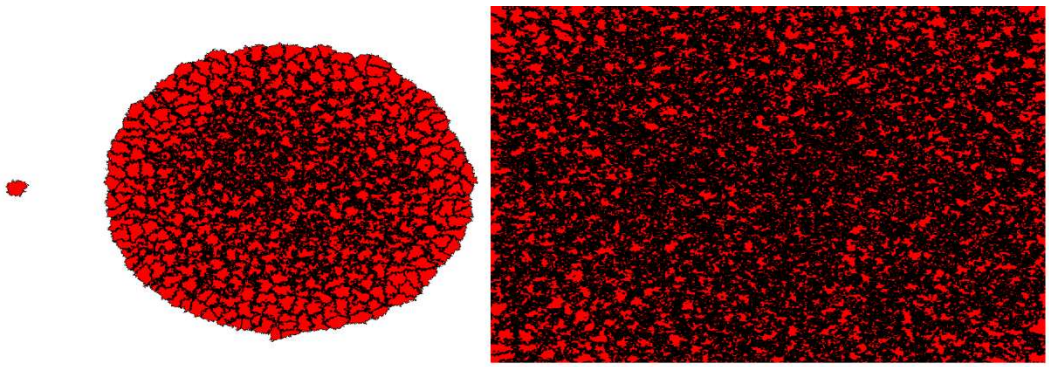


9. ábra: A sejtek által lefedett terület szimulációs idő függvényében.



10. ábra: Sejtek számának növekedése eltelt szimulációs idő függvényében.

Utolsó érdekességként pedig szeretném, ha megfigyelnék a szimulációról készült képernyőképeket. Ezeken látható, hogy a sejtcsoportosulás közepén jóval sűrűbb a sejtek száma, mivel a sejtek vonzzák egymást az adhéziós módosító miatt. Ebből kiindulva, próbálnak a sejtek egy „góccá” összeállni. Ha elég ideig futtatnánk a szimulációt, az összes sejt összeállna egy nagy sejt csoportosulássá.



11. ábra: Sejtek növekedésének ábrázolása a szimuláció közben. Megfigyelhető a sejt csoportok középpontjában történő aktívabb sejt folyamatok, amelyeket a sejtek falának a sűrűsége mutat be. Az első képen 1 darab sejt látható, a másodikon 368 db, az utolsón 708 db

6.3 Sebgyógyulás szimuláció

Utolsó szimulációként szeretnék végrehajtani egy olyan kísérletet, amelyet előttem már a valós életben is végrehajtottak. Így szeretném azt bizonyítani, hogy a modellem, illetve a programom használható a való életben is, illetve, hogy segítheti más kutatók munkáját.

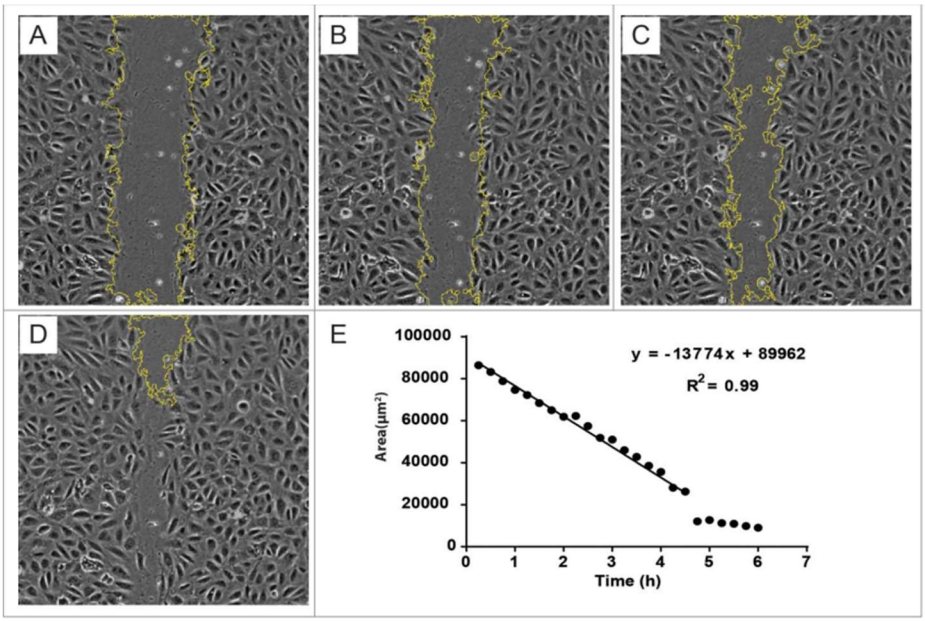
Választásom egy olyan kísérletre esett, amely a sejtek migrációját vizsgálja. A kísérlet során fognak egy sejtcsoportosulást, egy sebet ejtenek kémia, valamilyen fizikai eszköz vagy hő segítségével, majd megvizsgálják, hogy az adott sejt mennyi idő alatt tölti ki újra a teret. Ezzel meglehet vizsgálni egy sejt migrációs sebességét. Ez a fajta vizsgálat segítheti megérteni a rákos sejtek szaporodását, az embrió fázisban történő változásokat, illetve sérüléskor történő seb gyógyulásoknak a folyamatát is.

A valós életben a kísérlet órákon keresztül tart, emiatt vizsgálata nagyon nehéz. Szükség van mikroszkópokra, különleges megvilágításra, illetve különböző speciális kamerákra, amelyek segítségével magát a kísérletet megörökíthetik. Ezen kívül az is számít, hogy honnan szerezték be az adott sejttípust, mivel a sejttípusok között is lehetnek apró különbségek, befolyásolva a kísérlet eredményét.

12. ábra: Az eredeti kísérlethez szükség van mikroszkópra, Forrás: [14]

A saját számítógépes modellemben ezekre az eszközökre nincs szükség, illetve maga a kísérlet nem órákon, hanem perceken belül megy végbe.

Az eredeti kísérlet [14] eredményeképpen kapunk egy diagrammot, amely tartalmazza, hogy a különböző sejttípusok mennyi idő alatt fedték le az adott teret, illetve képeket arról, hogy a lefedés milyen sejtalakzatokat hozott létre.



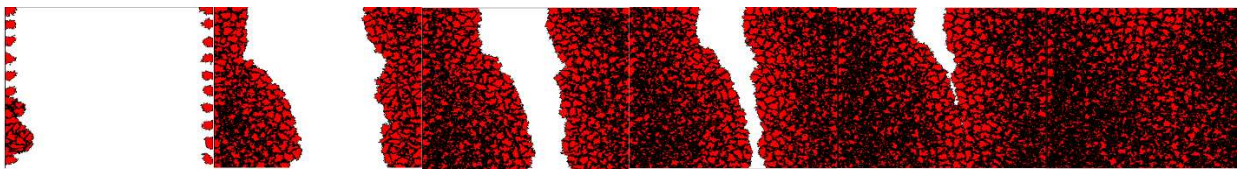
13. ábra: Az eredeti kísérlet eredményei, Forrás: [14]

A szimuláció előkészítését úgy kezdtem, hogy beállítottam egy 500 egység széles és magas teret. Ennek a térnek a két szélén elhelyeztem 10-10 darab sejtet, így szimulálva a sérült

középpontot. A sejtek azonos típusúak, maximális méretük 625 egység, 150 egységnyi növekedési paraméterrel, illetve az adhéziós paramétereket úgy állítottam be, hogy a környezet ne legyen hatással a sejtekre, ő maguk pedig vonzzák egymást.

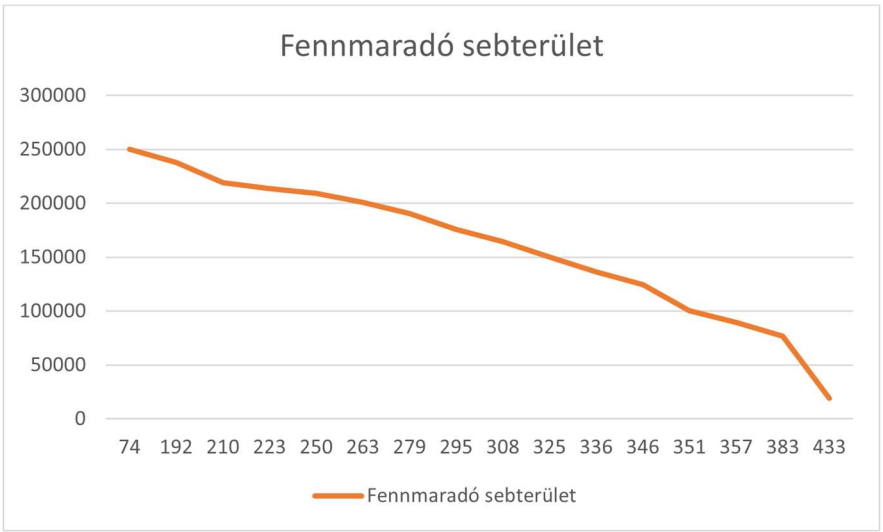
Ez a kísérlet a saját hardveres környezetemben 2 percet vett igénybe. A szimuláció pontossága állítható az eredeti sejtek számának növelésével, illetve a szimulációs tér növelésével, de mivel én most csak a működést szeretném bizonyítani, így megelégedhetünk ezekkel az eredményekkel is.

Összesen 429 darab sejt keletkezett és 433 szimulációs lépés hajtódott végre. A szimulációról készült képeken látható a végbement folyamat, minden lépése nyomon követhető. Látható, hogy a sejtek elkezdték kitölteni az üres teret mind szaporodással mind növekedéssel.



14. ábra: Én kísérletem megjelenése a programon belül

Ha pontosítjuk a bemeneti sejt paramétereket, akkor elérhetünk olyan sejtműködést, amely megegyezik az eredeti kísérletben szereplő sejtek migrációs sebességével, így szimulálva egy adott igazi sejtet a program segítségével.



15. ábra: Kísérletem által keletkezett diagram, amely ábrázolja a begyógyult területet idő függvényében

7. Továbbfejlesztési lehetőségek

7.1 Több Hamilton operátor implementációja

Alkalmazásomban időhiányból eredendően csak 4 darab operátor megvalósítására került sor. Ezek a módosítók képesek az alap sejtfunkciókat lefedni, úgymint szaporodás, növekedés, rendeződés, véletlenszerű mozgás, viszont a jövőben szeretnék több operátort is megvalósítani, hogy elérhessek bonyolultabb sejtfunkciókat is. Szeretném megvalósítani az irányított mozgásért használt operátort, így elérve a taxis sejtfunkciókat, illetve szeretném még lefejleszteni a sejthalálért felelős operátort. Jelenleg a szimulációmban nincs lehetőség arra, hogy a sejtek meghaljanak, így sok kísérletet nem lehet vele megvalósítani, amelyben szükség lenne erre a funkcióra. Ezentúl szeretném felvenni a kapcsolatot egy olyan személlyel, aki használná is az applikációmat a valós életbe, kikérve az ő véleményét, hogy milyen funkciók elengedhetetlenek még a sejtek teljes lefedéséhez.

7.2 Több kísérlet megvalósítása

Fontos lenne még több kísérlet megvalósítása is. Jelen dolgozatban csak egy igazi kísérletet valósítottam meg, hogy bizonyítsam a megvalósított alkalmazásom működését. A jövőben szeretnék több, komplexebb kísérletek után is nézni, amelyek végrehajthatóak a modell segítségével. Jelen világunkban szerintem két fontos kutatás van, amelyekhez segítő eszközként lehetne használni munkám: rákossejt kutatást, illetve a különböző vírusok fejlődésének a kutatása. A modell képes mindkettőre a megfelelő paraméterezéssel, kezdeti beállítással, viszont ezen beállítások megtalálása nagyon időigényes.

7.3 Importálás és exportálás különböző formátumokba

Jelenleg nincs lehetőség arra, hogy olyan kísérleti eredményeket töltsünk be az applikációmba, amelyek egy másik program által jöttek létre. Szeretném, ha lenne lehetőség arra, hogy azok az applikációk, amelyek jelenleg is használatban vannak sejtek kutatására, képesek legyen az én applikációmmal való kommunikációra. Így akár arra is lehetőség lenne, hogy igazoljam egy másik modell helyességét, vagy akár teszteljem az én modellem futását.

Ezentúl fontos lenne, hogy az én alkalmazásom is képes legyen ezekbe a szabvány formátumokba mentést végrehajtani, így az általam kapott eredményeket megoszthatnám különböző tudományos fórumokon.

Kivonat

A kísérletek számítógépen való végrehajtása, a hardverek, illetve fejlesztői eszközök fejlődésével egyre elterjedtebb, nagyon sok kísérlet előtt először számítógépen modellezik bizonyítva az eredeti feltevést.

A sejtbiológia egy összetett tudományág, nagyon sok kísérleti lehetőséggel. Ezek a kísérletek gyakran idő, eszköz, illetve tudás igényesek, azonban egy jól megvalósított számítógépen történő szimulálás kihasználásával ezek a hátrányok átugorhatóak. Így segítséget kaphatunk akár egy valós kísérlet megtervezéséhez, vagy olyan bonyolult folyamatok megfigyeléséhez, amelyekre nincs lehetőség a hagyományos módszerekkel.

A dolgozatom célja, hogy bemutassam milyen lehetőségek vannak a sejtek számítógépes szimulációjára, valamint egy saját modellt is létrehozok. Szeretném, ha az én megvalósításomban elérhető lenne a sejtek felparaméterezésére, illetve a kísérleti eredmények megtekintése alkalmazáson belül. Ily módon szeretnék egy kis betekintést adni a sejtek és a számítógépek világának a kapcsolatába.

Abstract

Computer-based experimentation is becoming more common with the advancement of hardware and development tools. Before many experiments, they are first modeled on a computer to prove the original assumption.

Cell biology is a complex discipline with many experimental possibilities. These experiments are often time-consuming, hardware-consuming, or knowledge-intensive, but these disadvantages can be overcome by taking advantage of a well-implemented computer simulation. In this way, we can get help either to design a real experiment or to observe complex processes that are not possible with traditional methods.

The aim of my dissertation is to show the possibilities of computer simulation of cells and to create my own model. I would like it to be available in my implementation to parameterize cells and view experimental results within an application. In this way, I want to give a little insight into the relationship between the world of cells and computers.

Hivatkozások

[1] D. M. Prescott, Cells: Principles of Molecular Structure and Function, Boston: Jones and Bartlett, 1988.

[2] A. Ghaffarizadeh, R. Heiland, S. H. Friedman és M. Shannon, PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems, 2018.

[3] M. Pranav, Å. Jan, W. Jan és K. Mikko, „CellSim3D," [Online]. Available: https://github.com/SoftSimu/CellSim3D. [Hozzáférés dátuma: 21 06 2021].

[4] F. Garner és J. A. James, „Simulation of biological cell sorting using a two-dimensional extended Potts model," *Phys. Rev. Lett.,* 1992.

[5] „CompuCell3D," [Online]. Available: https://compucell3d.org/FrontPage. [Hozzáférés dátuma: 11 03 2021].

[6] „Morpheus," [Online]. Available: https://morpheus.gitlab.io/. [Hozzáférés dátuma: 11 03 2021].

[7] F. R. Cooper, R. E. Baker, M. O. Bernabeu, R. Bordas, L. Bowler, A. Bueno-Orovio, H. M. Byrne, V. Carapella, L. Cardone-Noott, J. Cooper, S. Dutta, B. D. Evans, A. G. Fletcher és J. A. Grogan, „Chaste: Cancer, Heart and Soft Tissue Environment," *Journal of Open Source Software,* 2019.

[8] B. Stroustrup, A C++ programozási nyelv I, II., Budapest: KISKAPU KIADÓ, 2001.

[9] „OpenCL," [Online]. Available: https://github.com/KhronosGroup/OpenCL-Guide. [Hozzáférés dátuma: 11 04 2021].

[10] „Dear Imgui," [Online]. Available: https://github.com/ocornut/imgui. [Hozzáférés dátuma: 12 04 2021].

[11] DirectX. [Online]. Available: https://support.microsoft.com/hu-hu/topic/a-directx-leg%C3%BAjabb-verzi%C3%B3j%C3%A1nak-telep%C3%ADt%C3%A9se-d1f5ffa5-dae2-246c-91b1-ee1e973ed8c2. [Hozzáférés dátuma: 21 04 2021].

[12] „Git," [Online]. Available: https://git-scm.com/about. [Hozzáférés dátuma: 28 03 2021].

[13] M. Matsumoto és T. Nishimura, „Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator,” *ACM Transactions on Modeling and Computer Simulation,* Vol. 1. , pp. 3-33, 1998.

[14] J. E. N. Jonkman, J. A. Cathcart, F. Xu, M. E. Bartolini, J. E. Amon, K. M. Stevens és P. Colarusso, „An introduction to the wound healing assay using live-cell microscopy,” *Cell Adhesion & Migration,*Vol. 8. , pp. 440-451, 2014.

Ábrajegyzék