**SZAKDOLGOZAT**

**Virtuális Petri-csésze**

**OE-NIK** Hallgató neve: **Lex Tamás József**

**2022** Hallgató törzskönyvi száma: **T/005724/FI12904/N**

Tartalomjegyzék

[1. Összefoglaló 4](#_Toc102997940)

[2. Bevezetés – Probléma analízise 4](#_Toc102997941)

[2.1. Biológiai háttér 4](#_Toc102997942)

[2.1.1. Sejt felépítése 4](#_Toc102997943)

[2.1.2. Sejt funkciói 5](#_Toc102997944)

[2.2. Létező megoldások sejtek szimulálására 7](#_Toc102997945)

[2.2.1. Physicell 7](#_Toc102997946)

[2.2.2. CellSim3D 7](#_Toc102997947)

[2.2.3. Cellular Potts modell 7](#_Toc102997948)

[2.2.4. Chaste: Cancer, Heart and Soft Tissue Environment 8](#_Toc102997949)

[Megállapítások 9](#_Toc102997950)

[3. Tervezés 10](#_Toc102997951)

[3.1. A kutatás célja 10](#_Toc102997952)

[3.2. Szimuláció lépései 10](#_Toc102997953)

[3.2.1. Monte-Carlo lépés 11](#_Toc102997954)

[3.2.2. Implementálandó bemeneti paraméterek és Hamilton-operátorok 11](#_Toc102997955)

[3.3. Rendszer terv 13](#_Toc102997956)

[4. Eszközök kiválasztása 15](#_Toc102997957)

[4.1. Programozási nyelv 15](#_Toc102997958)

[4.2. GPU programozás 15](#_Toc102997959)

[4.3. Megjelenítés 15](#_Toc102997960)

[4.4. Verzió kezelés 15](#_Toc102997961)

[5. Elkészült szoftver bemutatása 17](#_Toc102997962)

[5.1 Párhuzamos Cellular Potts modell 17](#_Toc102997963)

[5.1.1 Adhéziós paraméterek 18](#_Toc102997964)

[5.1.2 Tömeg paraméterek 18](#_Toc102997965)

[5.1.3 Aktivitás paraméterek 18](#_Toc102997966)

[5.2 Megjelenítés és kommunikáció a videókártyával 19](#_Toc102997967)

[5.3 Felhasználói interfész bemutatása 19](#_Toc102997968)

[6. Kísérletek végrehajtása 20](#_Toc102997969)

[6.1 Egy sejt szimulálása 20](#_Toc102997970)

[6.2 Sejtek szaporodásának szimulálása 20](#_Toc102997971)

[6.3 Ising modell szimulációja 20](#_Toc102997972)

[6.4 Sebgyógyulás szimuláció 20](#_Toc102997973)

[7. Összegzés és tovább fejlesztési lehetőségek 20](#_Toc102997974)

[Kivonat 20](#_Toc102997975)

[Abstract 20](#_Toc102997976)

[Hivatkozások 20](#_Toc102997977)

# 1. Összefoglaló

Életünk legapróbb építő elemei a sejtek. Ezek a kis építő elemek szabják meg a hétköznapjainkat, belőlük épül fel az egész élővilág körülöttünk, 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. Mint az élet sok más területén az informatika itt is segítségünkre lehet, ugyanis sejtek számítógéppel való szimulációját már nagyon régóta próbálja a szakma végrehajtani.

Szakdolgozatom célja, hogy bemutassam ezen megoldásokat és, hogy elkészítsem a saját implementációm sejtek életciklusának a szimulálására. Az én szimulációm képes lesz több sejt valós idejű szimulációjára, amely a modellem párhuzamos megvalósítása által lesz lehetőség. Az így kapott eredményeket össze fogom hasonlítani a korábbi megoldásokkal, így kielemezve az én implementációm helyességét.

# 2. Bevezetés – Probléma analízise

## 2.1. Biológiai háttér

A probléma megértéséhez és a megoldás létrehozásához szükség van a sejtek definíciójára. Ebben a fejezetben egy rövid bevezetés fogok tenni a sejtekre. A sejtek tanulmányozásával egy külön tudományág a citológia [1] foglalkozik. Ennek a szakdolgozatnak nem célja a sejtek teljes bemutatása, ezért csak egy rövid áttekintőt készítek róluk.

### 2.1.1. Sejt felépítése

Minden élőlény aki a földön él sejtekből épül fel. Az apró baktériumoktól kezdve, a növényeken át, egészen az emberig minden élőlény ezekből az építőelemekből áll össze. Emiatt a sejtek nagyon változó felépítéssel, alakkal jöhetnek létre. Vannak akik egy sejtként alkotnak élő organizmust és vannak olyanok, amelyek több sejt kapcsolatából létrejött hálózatok segítségével alakítanak ki komplex működési formákat.

Általános mondhatjuk, hogy a sejtek két csoportba osztjuk. Vannak a **Prokarióta** és az **Eukarióta** sejtek. A prokarióta sejtek kevésbé komplexek, kevesebb DNS-t tartalmaznak és nincs elhatárolt sejtmagjuk. Minden prokarióta sejt egysejtű életmódot folytat. Ebbe a csoportba például a baktériumok tartoznak.

Többsejtű életformák eukarióta sejtekből épülnek fel. Az eukarióta sejtek nagyon sok -féle funkcióval rendelkeznek, egy szervezeten belül átlagosan több mint 200 fajta sejt különböztethető meg.

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. A DNS tekinthető a sejt tervrajzának, ez alapján képes újra alkotni önmagát. A sejtszervecskék tekinthetőek a sejt szerveinek, mindegyik különböző funkcióval rendelkezik és nélkülözhetetlen a sejt életében. A sejtet felépítő elemek között egy vízszerű anyag az úgy nevezett **sejtplazma** található.

A képen térkép látható

Automatikusan generált leírás

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 anyagcseréje miatt 9-10 óra alatt végrehajtja ezt a ciklust.

A sejtciklus általánosan 4 fázisra bontható. A G1 fázisban a sejt növekedik, 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 előröl, de létezik olyan sejt, amely képes megszakítani ezt a ciklust és osztódás után egy úgy nevezett G0 fázisba lép, amikor a sejt ellátja funkcióját, ám nem gyarapodik és nem osztódik.

A **sejtosztódás** folyamata során az anyasejtből két leánysejt keletkezik. Két fajta sejtosztódás van, a meiosis és a mitosis. Ebben a szakdolgozatban a mitosis folyamatával foglalkozom. Mitosis 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, és emiatt hibásan látja be funkcióját. Nem kontrollált mitosissal például a Tumor sejtek szaporodnak.

**Sejthalálnak** nevezzük az a folyamatot, amikor a sejt befejezi életfunkcióit. Ez történhet külső hatások miatt, vagy valamilyen belső vezérlés hatására. Programozott sejthalálnak nevezzük az Apoptózist, amely során a sejt lebontja önmagát utasítás hatására. Apoptózis fontos folyamata az életünknek, ugyanis ezzel a folyamattal kiszűrhetőek az „elöregedett” sejtek. Nekrózis során a sejt valamilyen sérülés vagy mérgezés (esetleg oxigén hiány) hatására pusztul el. 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 információk, hogy a rákos sejtek képesek meggátolni az apoptózis folyamatát, így úgymond sosem halnak meg.

A sejtek képesek egymással **kommunikálni** is. A kommunikációnak különböző formái vannak. Létezik olyan formája, amely a sejtek közvetlen kapcsolatát igényli. Ilyenkor a két sejt membránja össze ér és az anyagok cseréje ezen keresztül megy végbe. Létezik olyan formája, amikor a sejt valamilyen anyagot diffundál a környezetébe,és a körülötte lévő sejtek reagálnak erre az anyagra. Ez általában valamilyen toxin felfedezések fontos,hogy a sejt figyelmeztesse a körülötte lévőket a veszélyről. Ez a kommunikáció csak lokális környezetben működik. Nagy távon való kommunikációra a sejtek alkalmazzák a szinaptikus jelzés. Ezen módszer alkalmazásakor a sejt egy kis nyúlvány segítségével kommunikál a körülötte lévőkkel. Az emberi szervezetben például az idegsejtek ily módon kommunikálnak. Az utolsó módszer ,amely alkalmazható hosszú távú kommunikációra az amikor valamilyen hormont vagy anyagot juttat a véráramba a sejt és az az anyagot ez a közeg így képes messzire eljuttatni.

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 és 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 felsorolok párat:

* 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 CPM modellen alapuló megoldásokkal. A megalkotói egy Agent Based modellt hoztak létre, így kikerülve a Cellular Potts modellben bemutatott hálórendszert. A sejteket külön funkciókkal írják le, a térfogat és sejtek egymáshoz való tapadását is külön funkciókba szervezték. Eredetileg rákos megbetegedések kutatására hozták létre, ám az így kialakult rendszer nem csak a rákos sejtek szimulációjára képes. A program az anyagok diffúziójára, a BioFVM nevezetű könyvtárt használja, amelynek megalkotói megegyeznek a Physicell megalkotóival. Ez a modell képes realisztikusabb szimulációs eredményeket adni, viszont implementációja, bővítése nehezebb és hardveres igényes 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 Mathlabba, kép formátumba és XML formátumba.

### 2.2.2. CellSim3D

Ez [3] a program egy több egyetemet is összefogó koalíció útján jött létre. Ez a modell inkább a sejtek fizikai tulajdonságai 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 sikeresen van használva sejtek és más komplex biológiai rendszerek számítógépen való reprezentációjára. Számos alkalmazás sejtek szimulálását is ezen modell segítségével hajtja végre. (CompuCell3D, Morpheus)

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

Az modell 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 és tömegét.

A Hamilton-operátor módosításával, új paraméterek hozzáadásával módosítható az sejtek viselkedése, így például elérhető a kemotaxis és a sejtkommunikáció folyamata is.

A képen szöveg, keresztrejtvény látható

Automatikusan generált leírás

2. ábra: Cellular Potts modell mátrix reprezentációja

### 2.2.4. Chaste: Cancer, Heart and Soft Tissue Environment

A Chaste [5] 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 az Ágens típusú szimulációk, ahol a sejteket különböző részecskeként jelenítenek meg. Ebben a megvalósításban a biológiai és kémiai folyamatokat pontosan tudni kell, megértésük és tovább fejlesztésük sokkal nehezebb. Ezen fajta megvalósításnak az előnye, hogy pontos eredményeket kaphatunk vissza egy adott kísérletről.

A másik fajta megvalósítások valamilyen matematika módszert használnak fel a sejtek szimulálására. Ezek a modellek azért lettek elkészítve, hogy segítsék a biológiai folyamatok megértését, implementálásuk egyszerűbb és egészen pontos eredményt adnak vissza kísérletekről.

Biológiai tudásom hiányában és mivel informatika képzésen inkább a számítógépes megvalósítás a fontos, ezért a matematika modell megvalósítása mellett döntöttem. A létező modellek közül a Cellular Potts modellt fogom megvalósítani.

# 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 ezeket valós időben lefuttatni és megjeleníteni.

Az alkalmazásomnak rendelkeznie kell egy felhasználói interfésszel, amely lehetőséget ad egy 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 szolgáltatni a felhasználó felé.

Az alkalmazásom fontos célja még, hogy a szimuláció valós időben történhessen, ezért 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 0-s azonosítót a környezetnek szokták adni. A többi mező ami értékkel rendelkezik részt vesz a szimulációban. Az azonos számmal rendelkező mezők ugyan azt a sejttípust jelentik.

Az alap modell az adhéziós erőket és a sejtnek a belső térfogat változását veszi figyelembe. Az algoritmus figyeli az ebből eredendő energia változásokat 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. Ezek a szimulációk minden ismétlésre ugyan azt az eredményt adják. A sztochasztikus szimulációk során valamilyen véletlenszerűség be kerül a rendszerbe. Ezt a véletlenszerűséget jellemezhetjük valamilyen valószínűségi változóval.

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

### 3.2.1. Monte-Carlo lépés

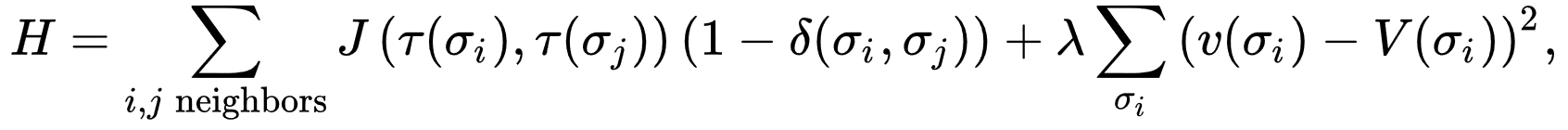
A Monte-Carlo algoritmus egy ismételt véletlen mintavételezésen alapuló megoldás. A folyamat során véletlenszerűen kiválasztunk egy mezőt majd ennek a mezőnek egy véletlenszerű szomszédját. Megpróbáljuk a mezőt átmásolni ebbe a szomszédos mezőbe. Kiszámítjuk az energia differenciát az így létre jött 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 számítást paraméterezhető egy T paraméterrel, amely a rendszer **Boltzmann-állandója** és tekinthető a rendszer hőmérsékletének is. Ez a megvalósítás nagyban hasonlít a szimulált lehűlés optimalizációs módszerre, amelyet Haladó Algoritmusok tárgy keretében tanultunk.

Egy adott rendszer energia tartalmának kiszámítására a Hamilton-operátorok összegét használjuk. A z 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.



*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”. Egy sejt tömegét az elfoglalt cellák összege adja ki. A második paraméter a λV pedig megadja, hogy egy adott szimulációs ciklus alatt mennyi növekedésre képed 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 való világhoz és a sejtek a környezetben lévő tápanyagok alapján képesek növekedni.

**Osztódási módosító** befolyásolja a sejt osztódási tömegét. 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** amely megadja a szimuláció hőmérsékletét. Minél magasabb ez az érték, annál aktívabb lesz a sejtek mozgása, mivel a negatív energia változással járó folyamatok nagyobb eséllyel mennek végbe.

## 3.3. Rendszer terv

Feladatom szoftver fejlesztési szempontból nem nagy kihívás, emiatt az UML diagramjaim nem lesznek nagyon látványosak. Az én projektemben az érték szerintem inkább magából a modell kiválasztásból, megvalósításából és párhuzamosításából ered. A Cellular Potts modell nagyban hasonlít a Haladó Algoritmusok tárgy keretében tanult Szimulált lehűlés technikára, ezért ebből indultam ki.

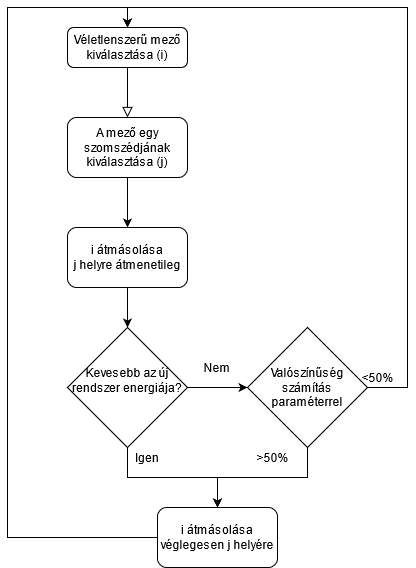
A képen szöveg látható

Automatikusan generált leírás

4. ábra: Szimulált lehűlés algoritmusa

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

13



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.

# 4. Eszközök kiválasztása

## 4.1. Programozási nyelv

A megvalósításom készítésekor sok szempontot figyelembe kell venni, de a fő szempont a gyorsaság nagy számítás igényű feladatok során. Választásom végül a C++ programozási nyelvre [6] esett, mivel támogatja a magas nyelvi funkciókat, de képes a memória alacsony szintű hozzáférésére is. Ezentúl képes a GPU programozás támogatására is és az ebből eredendő teljesítmény növekedés nélkülözhetetlen 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 [7] . 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.

## 4.3. Megjelenítés

//TODO:FIX

## 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 [9] esett korábbi tapasztalatok és a branching modell használata miatt. Git 2005ben 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. A Github egy webes szolgáltatásokat nyújtó alkalmazás 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 jobb teljesítmény elérés érdekében //TODO:LINK, emiatt 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 számítás, 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 néztem. A végső választásom a Mersenne Twister //TODO:FORRÁS nevezetű algoritmus megvalósítására esett. Ez a pszeudo-véletlen generátor sokkal jobb eloszlással rendelkezik mint az alap C++ által megvalósított verzió, emiatt a szimuláció-k 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 sejt-en és ha osztódás be van állítva, akkor azt is itt hajtom végre.

A megvalósított 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 textura, amit a rendereléskor 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 TODO:FORRÁS 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

# 6. Kísérletek végrehajtása

Ebben a fejezetben szeretnék bemutatni néhány kísérletet, amelyet végre lehet hajtani a programom segítségével. A program 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.

Bemutatok egy olyan kísérletet, amelyet a Cellular Potts modell helyességének a vizsgálatára lehet használni. Több megvalósított Cellular Potts modell tanulmányozásakor is észrevettem, hogy ez a kísérlet az elsők közt lett megvalósítva, mivel az eredménye látványos és így merhető, hogy a különböző módosítók jól működnek-e.

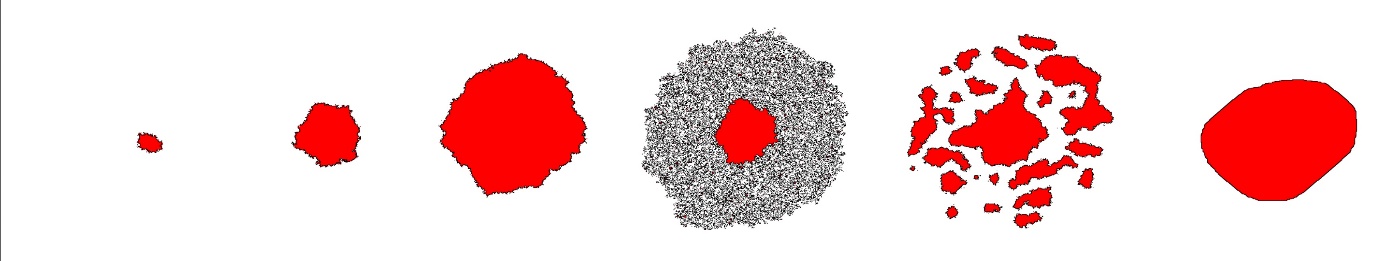
Végül szeretnék bemutatni egy olyan kísérletet, amely több sejtnek az egyidejű szimulációjával történik, ezzel bizonyítva, hogy az elkészült modellem használható akár olyan kísérletek számítógéppel való szimulációjára, amelyeket amúgy csak hagyományos labor körülmények között tudnánk végrehajtani. Fontos, 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 ezt a módosítót negatív irányba mozgatom, akkor pedig 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.



. ábra: Különböző sejtformációk a paraméterek hatásaira

## 6.2 Sejtek szaporodásának szimulálása

## 6.3 Ising modell szimulációja

## 6.4 Sebgyógyulás szimuláció

# 7. Összegzés és tovább fejlesztési lehetőségek

# Kivonat

# Abstract

# Hivatkozások

|  |  |
| --- | --- |
| [1] | ,. David M. Prescott, Cells: Principles of Molecular Structure and Function, Boston: Jones and Bartlett, 1988. |
| [2] | R. H. S. H. F. S. Ahmadreza Ghaffarizadeh, PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems, 2018. |
| [3] | J. Å. J. W. M. K. Pranav Madhikar, „CellSim3D,” [Online]. Available: https://github.com/SoftSimu/CellSim3D. [Hozzáférés dátuma: 21 06 2021]. |
| [4] | F. Graner és J. Glazier, „Simulation of biological cell sorting using a two-dimensional extended Potts model,” *Phys. Rev. Lett.,* 1992. |
| [5] | F. R. ,. R. E. M. O. R. L.-O. A. H. M. V.-N. L. J. S. B. D. A. G. J. A. Cooper, „Chaste: Cancer, Heart and Soft Tissue Environment,” *Journal of Open Source Software,* 2019. |
| [6] | B. Stroustrup, A C++ programozási nyelv I, II., Budapest: KISKAPU KIADÓ, 2001. |
| [7] | [Online]. Available: https://github.com/KhronosGroup/OpenCL-Guide. [Hozzáférés dátuma: 11 04 2021]. |
| [9] | [Online]. Available: https://git-scm.com/about. [Hozzáférés dátuma: 28 03 2021]. |
| [10] | D. R.Greening, „Parallel simulated annealing techniques,” *Physica ,* pp. 293-306, 1990. |

**Ábrajegyzék**

[1. ábra: Sejt felépítése 5](#_Toc103079457)

[2. ábra: Cellular Potts modell mátrix reprezentációja 8](#_Toc103079458)

[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 12](#_Toc103079459)

[4. ábra: Szimulált lehűlés algoritmusa 13](#_Toc103079460)

[5. ábra: Folyamatábra 14](#_Toc103079461)

[6. ábra 21](#_Toc103079462)