ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

SZAKDOLGOZAT

Virtuális petri-csésze

OE-NIK	Hallgató neve:	**Lex Tamás József**
2021	Hallgató törzskönyvi száma:	**T/005724/FI12904/N**

Tartalomjegyzék

1. Bevezetés

Az élet legapróbb építő eleme a sejt. Működésük megértése nélkülözhetetlen ahhoz, hogy válaszokat találjunk a létezés alapvető kérdéseire. A sejtek befolyásolják mindennapi életfolyamatainkat, fejlődésünket vagy akár gyógyulásunkat is. Abban az estben, ha képesek lennénk megértésükre, befolyásolhatnánk őket és egy új fejezetet nyithatnánk a modern orvostudományban. Az eddig gyógyíthatatlannak tűnt betegségek gyógyíthatóak lennének. Végtag vesztés esetén az ember növeszthetne egy újat, szervátültetés helyett beindíthatnánk a sejtosztódást és újat növeszthetnénk a szervezeten belül, vagy a legfontosabb és a legtöbbet vizsgált eset, hogy a rák is gyógyítható lenne.

Az életformák megjelenhetnek szimpla egysejtű vagy akár többsejtű formában is. A többsejtű rendszerek folyamatos sejtosztódás folyamatával jönnek létre. Példaként az embereket felépítő többsejtű rendszerek egyedszáma meghaladja a 100 billiót, ahol minden egyes sejtnek külön funkciója van. Sejtosztódás egy bonyolult több lépcsős folyamat, amely során a szülősejtből két leánysejt keletkezik. Ennek a folyatnak a vizsgálata kritikus pont a sejtek megértésében. A sejtek osztódását legelőször 1835-ben figyelte meg Hugo von Mohl német botanikus mikroszkópja segítségével. A számítógépek megjelenésével és elterjedésével igény merült a sejtek ily módon történő megfigyelésére és szimulációjára. A sejtek szimulációja a folyamat bonyolultsága és az egyedek száma miatt nagy erőforrás igénnyel rendelkezik, ezért a mai napig egy teljesen biológiailag pontos sejt ciklus szimulációjára még nem volt példa.

2. 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. [1] A sejtek tanulmányozásával egy külön tudományág a citológia 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.

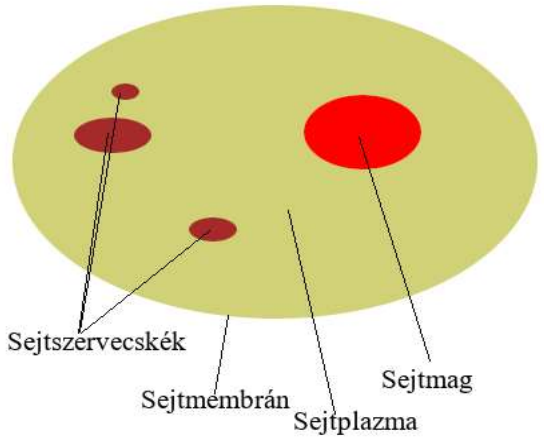
Általános mondhatjuk, hogy a sejtek két csoportba osztjuk. Vannak a **Prokarióta** és az **Eukarióta** sejtek. A prokarióta sejtek kevésbé komplexek, kevesebb DNS-t tartalmaznak és nincs elhatárolt sejtmagjuk. Minden prokarióta sejt egysejtű életmódot folytat. Ebbe a csoportba például a baktériumok tartoznak.

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 replikálni 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.

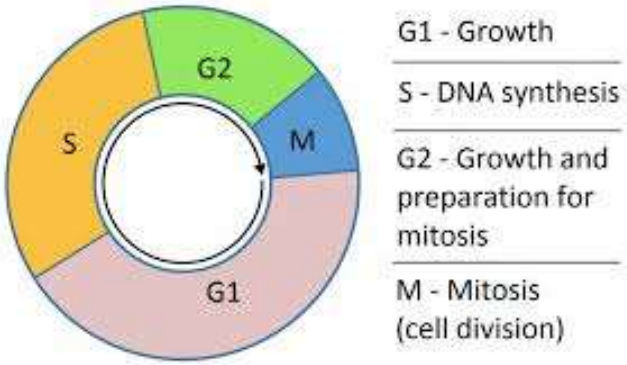


1. Ábra: Sejt felépítése

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.



2. Ábra: Sejtfázisok

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:

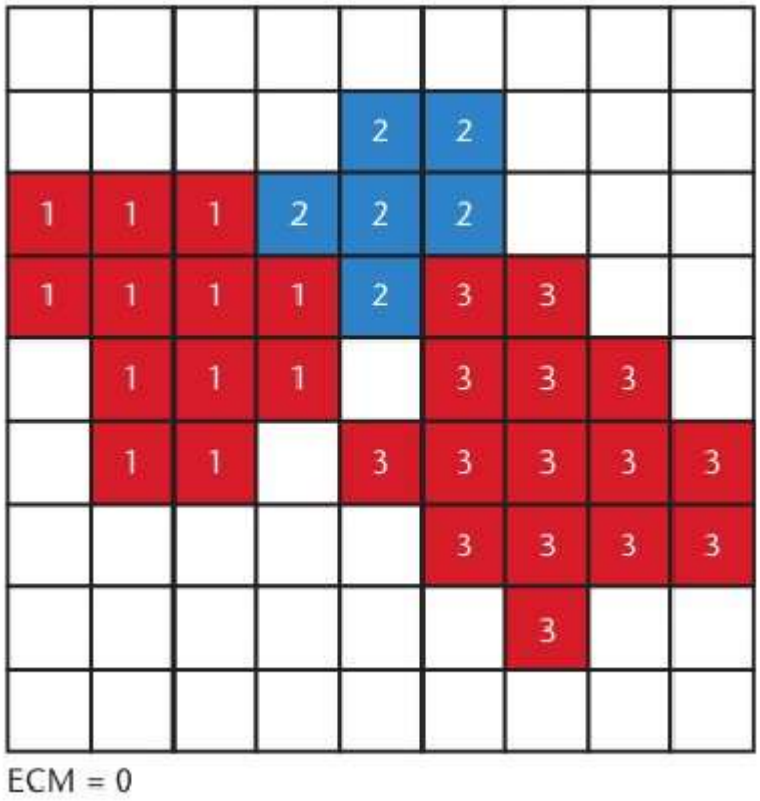
- Kemotaxis: reakció valamilyen vegyi anyagra (vonzás vagy taszítás toxin esetén)
- Fototaxis: mozgás a fény irányába, fény intenzitásától függően
- Termotaxis: mozgás a hőmérsékleti gradiens irányába
- Aerotaxis: reakció a környezet oxigén koncentrációjára

2.2. Létező megoldások bemutatása

2.2.1. Cellular Potts modell

Első létező megoldásként szeretném bemutatni a **Cellular Potts modell**-t.[2] Ez a 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 sejtet ábrázol. Az azonos azonosítóval 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.



3. Ábra: Cellular Potts modell ábrázolása

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 egy Hamilton-operátort használ. Ez 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.

2.2.2. Physicell

[3]Physicell 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.

2.2.3. CellSim3D

[4]Ez 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.

3. Eszközök kiválasztása

3.1. 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 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.[5]

3.2. 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. A CUDA csak az Nvidia videokártyák által van támogatva, az OpenCL [6] 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 (pl.: 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.

3.3. Megjelenítés

Az alkalmazásom célja az is, hogy a szimuláció eredményét valamilyen formában prezentáljam. Ehhez szükségem lesz valamilyen megjelenítésre és felhasználó interfészre. Ennek létrehozására a QT nevezetű eszközre esett a választásom.[7] QT egy

keresztplatformos felhasználó interfész keretrendszer, amely segítségével több platformra is lehetséges ugyan azzal a kódbázissal készíteni alkalmazásokat. Azért erre a keretrendszerre esett a választásom, mivel a QT nyílt forráskódú alkalmazások készítése esetén ingyenes és mivel sok olyan alkalmazás amelyet a hétköznapjainkban is használunk ezzel a keretrendszerrel készült.

3.4. 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 [8]GIT-re 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.

4. Tervezés

A programom tervezése során először egy Agent alapú rendszer készítésével próbálkoztam, ám az első prototípusok létrehozásakor sok hibába és nehézségbe ütköztem. Ezek a rendszerek nagyon közel állnak a valósághoz, ám létrehozásuk sok tudást és időt igényel. Ezek a nehézségek miatt kezdtem el kutatni újabb megoldások után, és ekkor találtam rá a fent is megemlített Cellular Potts [9][10] modellre. Választásom végül ennek a modellnek a megvalósítására esett, mivel az ezen módon futtatott sejt szimulációk is nagyon közel állnak a valósághoz, a megvalósításuk jól párhuzamosítható, és a sejtek viselkedése jól befolyásolható a Hamilton-operátor módosításával.

4.1. Pontos specifikáció

Szakdolgozatom célja egy olyan alkalmazás tervezése, megalkotása és tesztelése, amely képes a sejtek szimulációjára. A sejteket a szimuláció előtt paraméterezhetni lehet, meg lehet adni, hogy milyen funkciókra képes majd ezek után a szimuláció során képes ezeket valós időben lefuttatni és prezentálni.

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 az őt használó felhasználó felé.

Az alkalmazásom fontos célja még, hogy a szimuláció valós időben történhessen, ezért szükség van a szimuláció párhuzamos megvalósítására.

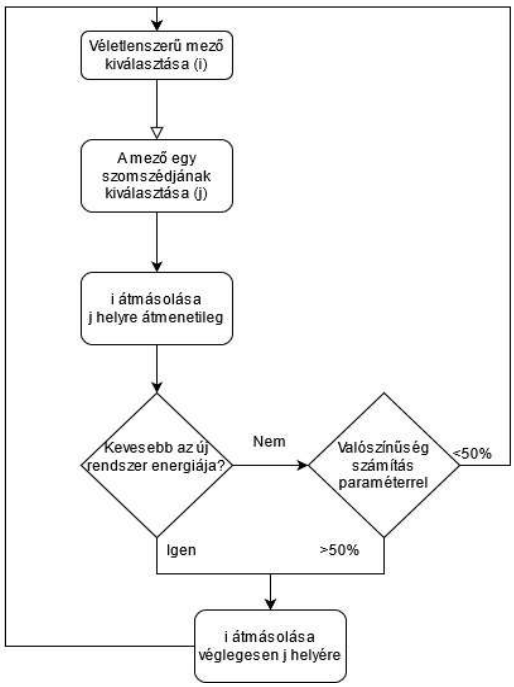
4.2. Szimuláció lépései

A CPM egyik nagy előnye, hogy szimulációs lépései jól definiáltak. Először is létre kell hozni egy hálót, és minden egyes elemének adni kell egy azonosító számot. A 0-s azonosítót a környezetnek szokták adni, ez az érték jelenti a „semmit”. Az azonos azonosítóval rendelkező mezők ugyan azt a sejtet jelentik.

Az alap CPM 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.



4. Ábra: Szimuláció folyamatábrája

4.2.1. Monte-Carlo algoritmus

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.4.2.2. Hamilton-operátor és módosítói

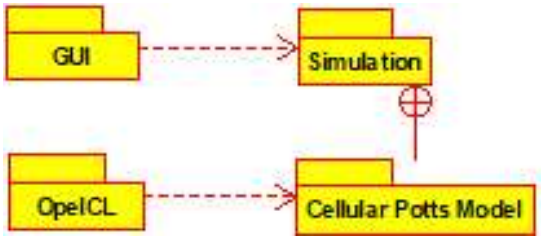
Egy adott rendszer energia tartalmának kiszámítására a Hamilton-operátor használjuk. A z alap modellben szereplő egyenlet figyelembe veszi az adhéziós erőt az azonos sejttípusok között,az adhéziós erőt a különböző sejttípusok között és végül a sejtnek a térfogatát és annak változásait.

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

5. Ábra: Alapértelmezett Hamilton-operátor

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.

4.3.Egyszerű rendszer terv

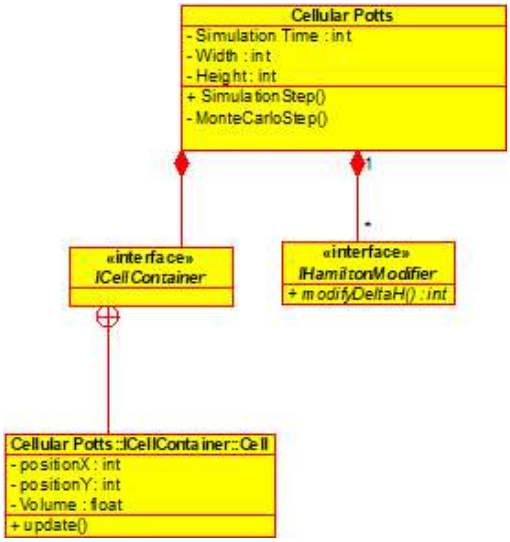


6. Ábra: Csomag ábra

Amint a csomag ábrából láthatjuk az alkalmazás két fő modulból fog felépülni. A GUI ami tartalmazza a felhasználói interfész implementációját és a Simulation amely tartalmazza a Cellular Potts modell megvalósítását és az OpenCL kommunikációt.

A felhasználói interfész a szimulációs modul adataiból dolgozik és azokat jeleníti meg. Magát a felhasználói interfészt MVC fejlesztési modell alapján fogom elkészíteni. Az .MVC modell az azt jelenti, hogy az megjelenítő alkalmazáson belül a 3 réteg lesz, a modell, a nézet és a vezérlő. Ily módon képes leszek a QT designer-el megtervezni egy nézetet, majd a vezérlő logikát a saját alkalmazásomban implementálni.

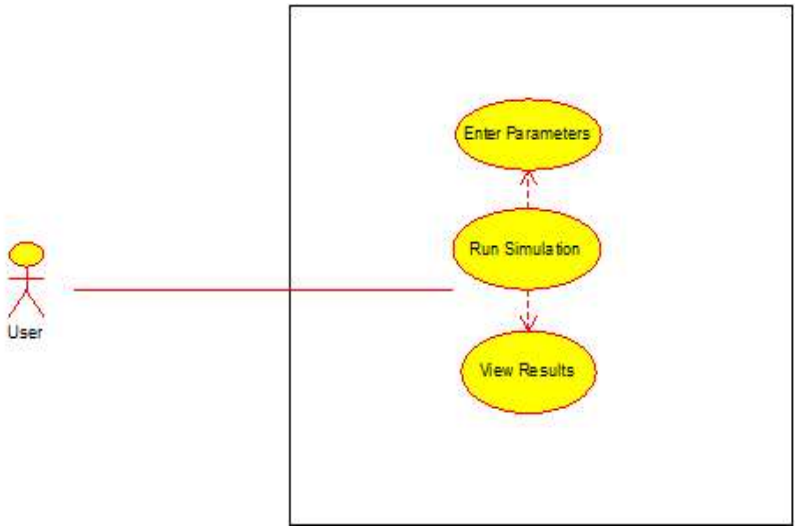
A Cellular Potts Modell része a szimulációnak, mivel a modell lépéseit valósítjuk meg. Legelőször ezt a szimulációt fogom implementálni. A szimuláció képes lesz megjelenítés nélkül is futni, így elősegíteni azt, hogy a későbbiekben elosztott rendszereken is implementálható legyen, ahol annyira nem fontos a megjelenés.



7. Ábra: Cellular Potts Modell diagram

Az fentebb lévő ábrán a Cellular Potts modell osztály diagramját lehet megtekinteni. A diagram ábrázolja,hogy a modell implementálja az algoritmus lépéseket, tartalmazza a sejteket és a Hamilton-operátor módosítókat.

Az alábbi Use-Case diagram ábrázolja a felhasználó lehetőségeit. A felhasználónak lehetősége van egy szimulációt elindítani, megadni a paramétereit, majd megtekinteni a szimuláció eredményét.



8. Ábra: Use Case diagram

Irodalomjegyzék

1: David M. Prescott, Cells: Principles of Molecular Structure and Function, 1988
2: Marco Tektonidis , Haralambos Hatzikirou, Arnaud Chauvière, Matthias Simon, Karl Schaller, Andreas Deutsch, Identification of intrinsic in vitro cellular mechanisms for glioma invasion , 2011
3: Ahmadreza Ghaffarizadeh, Randy Heiland, Samuel H. Friedman, ShannonM. Mumenthaler, Paul Macklin, PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems, 2018
4: Pranav Madhikar, Jan Åström, Jan Westerholm, Mikko Karttunen, CellSim3D: GPU accelerated software for simulations of cellular growth and division in three dimensions, 2018
5: Bjarne Stroustrup, A C++ programozási nyelv I, II., 2001
6: https://github.com/KhronosGroup/OpenCL-Guide, utoljára megtekintve: 2021.04.08.,
7: https://www.qt.io/, utoljára megtekintve: 2021.04.11.,
8: https://git-scm.com/about, utoljára megtekintve: 2021.03.29.,
9: Marco Scianna, Luigi Preziosi, Cellular Potts Models, 2013
10: Marco Scianna, Luigi Preziosi, A Cellular Potts Model simulating cell migration and in matrix environments, 2013

Ábrajegyzék