Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство
Ниво студија: Мастер академске студије
Модул: Примењена механика и аутоматско управљање
Предмет: Компјутерска графика
Број индекса: 416/2017

Ана М. Вуловић

Моделирање раста мигрирајућих ћелија посредством ћелијског аутомата

Мастер рад

Комисија за преглед и одбрану:

1. **Проф. др, Ненад Филиповић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

Изјављујем да је Мастер рад, Формални оквир управљања комуникацијом са аспекта заштите интелектуалне својине, као и ауторских права на високошколским и факултетским институцијама у оквиру територије републике Србије, резултат мог, сопственог рада, а да су библиографске референце јасно наведене и сумиране у оквиру Литературе.

Захваљујем се ментору проф. др Ненаду Филиповићу на корисним саветима, пруженој подршци и примедбама током израде овог рада.

Кандидат:

Вуловић Ана

Садржај

Увод 2
1.Васкулогенеза и ангиогенеза 4
1.1 Ендотелне ћелије 4
1.2 Покретљивост ендотелних ћелија 5
1.2.1 Покретљивост индивидуалних ћелија 7
1.2.2 Колективна миграција ћелија 7
2. BPAE ћелије 7
3. Ћелијски аутомати 9
2. Дводимензиони ћелијски аутомат 11
5. Ћелијски аутомат за моделирање миграционих ћелија са контактном инхибицијом 15
5.1 Алгоритам ћелијског аутомата за симулацију миграторних ћелија са контактном инхибицијом 17
5.1.1 Рутина за промену смера: 18
5.1.2 Рутина за деобу ћелије: 18
5.2 Имплементациони детаљи ћелијског аутомата 18
6. Резултати и дискусија 28
Закључак 39
Референце 40
Прилог 1 – Једнодимензиони ћелијски аутомат 42
Прилог 2 – Дводимензиони ћелијски аутомат 43
Прилог 3 – Ћелијски аутомат за симулацију пролиферације миграторних ћелија са контаткном инхибицијом 45

Списак скраћеница

CA – cellular automaton
BPAE – Bovine pulmonary artery endothelial
CIL - Contact inhibition of locomotion
bFGF - Basic fibroblast growth factor
ECM - extracellular matrix
PDFG - platelet-derived growth factors
TGFβ - transforming growth factor β
ACE – angiotensin converting enzyme

Списак слика

Слика 1 Попречни пресек крвног суда са назначеним ендотелом.....5
Слика 2 Шема ћелија које мигрирају на дводимензионалним (2D) или 3D матрицама. Пример 2D показује принципе механобиологије помоћу којих ћелија чита (зелене стрелице) механичка својства ECM-а и претвара их у биохемијски унутарћелијски сигнал (црвена стрелица) који утиче на цитоскелет, сигнализацију и транскрипцију. На крају, ћелија реагује и применом сила на саму матрицу (плава стрелица) и подвргавањем процесима као што су пролиферација, апоптоза (ћелијска смрт), диференцијација и миграција (велика плава стрелица). [10].....6
Слика 3 Фон Нојманова (а) и Мурова околина (б) [13] 9
Слика 4 Симулација једнодимензионог ћелијског аутомата у првих 30 временских корака 11
Слика 5 Чести шаблони који се појављују у Игри Живота [15] 13
Слика 6 Дијаграм стања једне ћелије у Игри Живота.....13
Слика 7 Резултат извршавања Игре Живота у временским корацима 1 (а) и 10 (б).....14
Слика 8 Трајекторија миграторних ћелија током 36 сати [17].....15
Слика 9 Хистограм времена чекања за различите правце ћелије (а) стационирана ћелија, (б-и) смерови ћелије [17].....15
Слика 10 Хистограм дистрибуције насумичних бројева позивањем методе за инверзну експоненцијалну дистрибуцију 20
Слика 11 Углови поља у околини за смер север (1).....22
Слика 12 HSB простор боја. Угао на кружници (H) представља нијансу боје, растојање од центра кружнице (S) представља засићеност, а висина (V) 27
Слика 13 Симулација пролфиерације непокретних ћелија, почевши од горе лево у смеру казаљке а) Иницијална колонија б) Колонија после 5 дана в) колонија после 25 дана г) колонија после 40 дана 29
Слика 14 График раста популације за непокретне ћелије 30
Слика 15 Симулација пролиферације ћелија са просечном брзином 3.5 микрометара по сату а) Иницијална колонија б) Колонија после 5 дана в) Колонија после 15 дана г)Колонија после 25 дана 31
Слика 16 Функција раста ћелија у односу на време, просечна брзина 3.5 микрометара по сату 31
Слика 17 Функција раста колоније ћелија у зависности од времена, брзина 28 микрометара по сату.....32
Слика 18 Симулација пролиферације ћелија са брзином 28 микрометара по сату а) иницијална колонија б) после 5 дана в) после 8 дана г) после 10 дана 32
Слика 19 Симулација пролиферације ћелија са брзином 56 микрометара по сату а) иницијална колонија б) после 2 дана в) после 6 дана г) после 8 дана 32
Слика 20 Функција раста колоније ћелија у зависности од времена, брзина 56 микрометара по сату.....33
Слика 21 Фунцкија раста колонија ћелија у зависности од времена и брзина [1].....33
Слика 22 Симулација пролиферације ћелија са просечном брзином 3.5 микрометара по сату и насумичном поставком иницијалне популације а) Иницијална колонија б) Колонија после 5 дана в) Колонија после 15 дана г) Колонија после 20 дана.....34
Слика 23 Функција раста колоније ћелија у зависности од времена, брзина 56 микрометара по сату, насумична поставка иницијалне популације.....34

Слика 24 Зависност просечне брзине популације у односу на време, брзине ћелија приказане су у легенди 35
Слика 25 Просечне брзине популације, добијене у раду [1] 35
Слика 26 Просечна брзина у односу на величину популације, насумични иницијални распоред 36
Слика 27 Просечна брзина у односу на величину популације, иницијални распоред концентрисан у средини 36
Слика 28 Просечна брзина у односу на величину популације, добијена у раду [1] 37
Слика 29 Зависност брзине популације у односу на величину иницијалне популације 37
Слика 30 Зависност брзине популације у односу на величину иницијалне популације, добијена из рада [1] 38

Резиме

У овом раду, приказан је значај пролиферације миграторних ћелија са контактном инхибицијом. Показан је математички апарат који је погодан за симулацију феномена раста ових ћелија у виду ћелисјке аутомате, дата је основа ћелијских аутомата као механизма за симулацију природних појава и развијен је симулатор пролиферације миграторних ћелија. Приказани су резултати симулације пролиферације са различитим брзинама миграције и почетним поставкама локације ћелија.

Кључне речи:

дискретно моделирање, контактна инхибиција, ћелијски раст

Abstract

In this thesis, the importance of proliferation of migratory cells with contact inhibition is presented. A mathematical apparatus suitable for simulating the phenomenon of growth of these cells in the form of cellular automata is shown, the basis of cellular automata as a mechanism for simulation of natural phenomena is given and a simulator of migratory cell proliferation is developed. The results of the proliferation simulation with different migration rates and initial cell location settings are presented.

Key words:

discrete modeling, contact inhibition, cell growth

Увод

Ћелијски аутомати налазе примену у многим гранама науке као средства при моделирању, међутим због специфичности и самих правила овог моделирања поставља се као један од најоптималнијих начина решавања задатог проблема ћелијске пролиферације. У овом раду извршиће се валидација постојећих експерименталних података [1] и симулирања раста ендотелних ћелија плућне артерије у случају експериментално испитиваном код говеда (BPAE cells) [1].

Значај у посматрању раста ове посебне врсте ћелија лежи у њиховим специфичним механизмима обнављања при повредама ендотелних структура. Ћелијска миграција је централни процес у развоју и опстанку вишећелијских организама. Формирање ткива за време развоја ембриона залечивање рана и одговори имуног система захтевају организовано кретање ћелија у специфичним правцима на одређене локације. Грешке при овим процесима доводе до озбиљних последица: интелектуалног онеспособљавања, метастаза, васкуларних болести и формације тумора [3]

Симулирање миграције ове врсте ћелија најлакше је извршено коришћењем ћелијског аутомата (у даљем тексту CA) узимајући у обзир сет правила и понашања сваке појединачне ендотелне ћелије и математичким описивањем њихове деобе. Битна особина при моделирању ендотелних ћелија описаниих у овом раду јесу особине које испољавају при контакту са околним ћелијама при чему се успорава кретање у контакту са суседном ћелијом (контактна инхибиција кретања - CIL). Овај механизам заустављања раста омогућава спречавање пролиферације односно продирања када дође до формирања непрекидног конфлуентног слоја ћелија у случају нормалних ћелија. У случају ћелија канцера, овај феномен је посебно битан јер ћелије канцера не поседују могућност заустављања пролиферације ни контактне инхибиције локомоције након доласка у контакт са суседном ћелијом.

Ова врста ћелија је јако битна у формирању и залечивању рана насталих у кардиоваскуларном систему, због чега је испитивање рађено управо на плућној артерији говеда, јер је унутрашњост артерије прекривена слојем ендотелних ћелија које играју значајну улогу при обнављању и срастању оштећених делова артерије. Кроз васкуларни систем, ендотелне ћелије задржавају способност за поделу и кретање. Многа истраживања [8] пружају доказе да одговор на повреду ендотела може довести до тромбозе и абнормалне пролиферације глатких мишића као резултат стимулисаног раста добијеног од тромбоцита на местима на којима је поремећен ендотелни континуитет.

Јер и тромбоза и ненормалан раст глатке мишићне ћелије су компоненте формирања лезије атеросклерозе [1], раст ендотела је неопходан поступак поправке за надокнађивање изгубљених ћелија као последице ћелијских механизама одумирања. Ендотелне ћелије, као и многе нормалне ћелије сисара, захтевају подлогу на којој се могу ширити и расти. Екстрацелуларни матрикс игра пресудну улогу у регулисању облика, поларитета и раста ћелија. Раст ендотелних ћелија (за разлику од фибробласта, ћелија

глатких мишића, или других ћелија), карактерише стварање високо уређеног конфлуентног моносло ја у коме се суседне ћелије додирују. Ако током овог процеса ћелија постане окружена другим ћелијама, престаће да расте. У овом раду је представљен модел раста и деобе ендотелних ћелија плућне артерије (BPAE) према параметрима добијеним из експеримента у раду [1]. Симулација је вршена посредством ћелијског аутомата и коришћени су експериментални резултати из литературе у циљу да се изврши валидација података што је у наставку рада и показано.

1.Васкулогенеза и ангиогенеза

Крвни судови су најранији и најједноставнији систем деловања ембриона који функционише. Васкуларне мреже се формирају тако рано у ембриогенези да истражитељи имају разуман приступ узорцима помоћу којих могу да посматрају и разумеју настанак васкуларног узорка, његове структуре и повезане функције. Даље, састављање и преуређивање васкуларног система је од виталне важности код вишеструких патологија - главни примери су зарастање рана и напредовање тумора. Како се васкуларне мреже састављају, основни је проблем развојне биологије који такође има медицински значај. Да бисмо објаснили организационе принципе који стоје иза васкуларног узорковања, морамо разумети како се структуре на нивоу ткива могу контролисати кроз обрасце понашања ћелија попут покретљивости и адхезије који су, пак, одређени процесима биохемијског преношења сигнала. Сви ови процеси имају обрасце, па ендотелне ћелије могу самостално да формирају међусобно повезану структуру. Наведени механизми важни су посебно код ендотелних ћелија чије су кретање и деоба симулирани у наставку рада помоћу ћелијске аутомате. [5]

1.1 Ендотелне ћелије

У општем смислу можемо издвојити неколико типова ћелија које сачињавају сва ткива у људском организму, фокус ове валидације јесте на проучавању ћелија које учествују у васкулогенези и репарацији, односно у изградњи ткива крвних судова, а то су ендотелне ћелије.

Ендотелне ћелије покривају унутрашњост крвних и лимфних судова, формирајући медијум између циркулације крви и лимфе у лумену и остатку зида крвног суда. Ове ћелије служе као пропусна баријера за регулацију протицања материја између крвотока и околног ткива. У базичним истраживањима, ендотелне ћелије су од велике важности у срастању рана, ангиогенези, можданој крвној баријери, запаљенским процесима, дијабетесу и другим кардиоваскуларним болестима.

Проблем се јавља при оштећењу ендотелног слоја унутар крвног суда при чему у покушају крвног суда да дође до репарације зида ендотела долази до низа реакција између ћелија и претходно здрав ендотел постаје оштећен услед губљења одређених својстава при дељењу и кретању ћелија (Слика 1.).

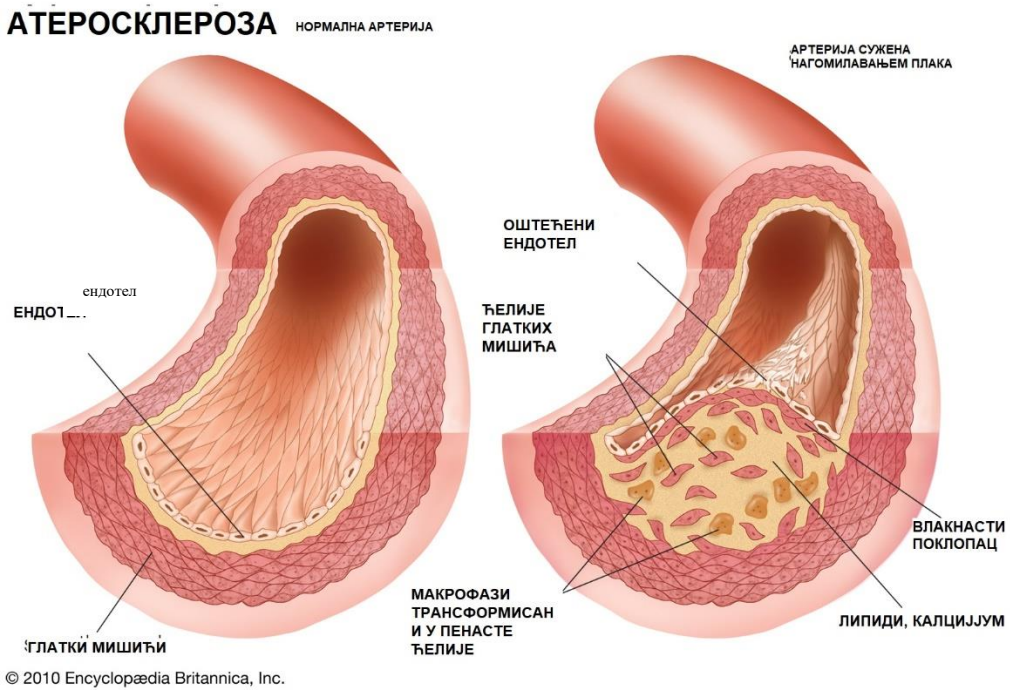
Како раст ендотелних ћелија (за разлику од фибробласта, ћелија глатких мишића, или других ћелија), карактерише стварање високо уређеног моносloја у коме се суседне ћелије додирују. Ако током овог процеса ћелија постане окружена другим ћелијама, престаће да расте. Као резултат ове појаве такозване - инхибиције контакта, само делић одрживих ћелија велике популације наставиће да расте и да се дели након почетних фаза овог процеса.

Овај делић одрживих ћелија опада са повећањем густине ћелија и у зависности је од покретљивост ћелија и почетних услова ћелијских култура [1]. Примећена је значајна покретљивост ендотелних ћелија под условима (као што је зарастање рана) који излажу

ћелије факторима раста. У раду [9] открили су да када се направи рез у ендотелном слоју аорте говеда, ћелије почну брзо да се крећу са ивице ране у огољену област. Ова миграција је регулисана основним фактором раста фибробласта (Basic fibroblast growth factor - bFGF) који су ослободиле саме ћелије. Ови резултати су показали да је bFGF одговоран за појачану покретљивост и повећане стопе пролиферације ендотелних ћелија.

У општем смислу постоји више подела ендотелних ћелија на:

- Ендотелне ћелије пупчане вене (Human Umbilical Vein Endothelial Cells (HUVECs))
- Ендотелне ћелије коронарних артерија (Coronary Artery Endothelial Cells (CAECs))



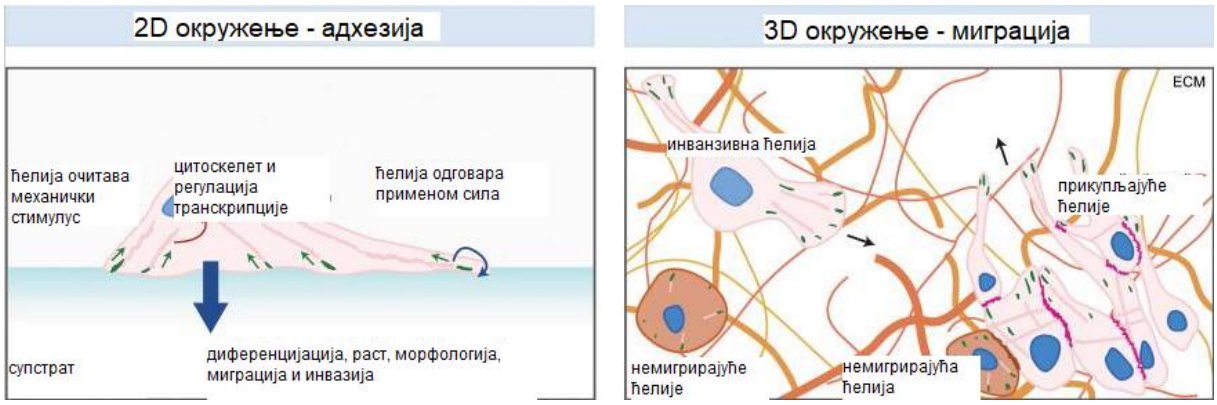
Слика 1 Попречни пресек крвног суда са назначеним ендотелом

- Ендотелне ћелије аорте (Aortic Endothelial Cells (AECs))
- Плућне микроваскуларне ендотелне ћелије (Human Lung Microvascular Endothelial Cells (HLMECs))

1.2 Покретљивост ендотелних ћелија

Као једну од централних особина ове врсте ћелија у раду, посматраћемо способност миграције и деобе ендотелних ћелија као и њихову коорелацију. Миграција ћелија је од круцијалног значаја за организацију ткива током развоја. Уколико дође до ремећења неког од механизама миграције ћелија последице по развој ткива и организме су велике. Грешке у миграцији и деоби ћелија доводе до озбиљних озлеђења али и неконтролисаног раста ткива, развоја тумора и канцера [7]. Једна од врста миграторних ћелија јесу и ендотелне ћелије. Ћелије, ткива и органи морају се стално прилагођавати свом

окружењу. Интеракција ћелије са околином је пресудна за физиолошку организацију ткива и функције током развоја, као и за хомеостазу, регенерацију и старење. Такође је укључена у патолошка стања - на пример, током прогресије тумора или фиброзе. Ћелијско микроокружење састоји се од суседних ћелија екстрацелуларног матрикса (ECM) и околног међућелијског медија. Микроокружење варира у саставу и организацији, у зависности од ткива или услова културе ин витро. На ћелијском нивоу, када ћелија додирне дозвољену површину, било да је подлога или нека друга ћелија, формираће лепљиве структуре које јој омогућавају да осети и реагује на својства свог окружења. Ћелије могу да осете две главне врсте информација: хемијске сигнале, попут малих молекула и растворљивих фактора, који се читају кроз одређене рецепторе, и физичка својства, укључујући крутост подлоге, топологију, порозност и еластично понашање, као и силе притиска и вуче. [10]



Слика 2 Шема ћелија које мигрирају на дводимензионалним (2D) или 3D матрицама. Пример 2D показује принципе механобиологије помоћу којих ћелија чита (зелене стрелице) механичка својства ECM-а и претвара их у биохемијски унутарћелијски сигнал (црвена стрелица) који утиче на цитоскелет, сигнализацију и транскрипцију. На крају, ћелија реагује и применом сила на саму матрицу (плава стрелица) и подвргавањем процесима као што су пролиферација, апоптоза (ћелијска смрт), диференцијација и миграција (велика плава стрелица). [10]

Интеракција између ћелије и њене околине функционише узајамно. С једне стране, ћелије осећају сигнале из околине (слика 2). Ћелије очитавају физичке стимулусе употребом механичких сензора. То се може догодити отварањем канала, истезањем протеина, излагањем места критичног везивања или индуковањем биохемијских сигналних путева. На крају су ове информације интегрисане тако да ћелија може на одговарајући начин да реагује. До одговора може доћи брзо преуређивањем цитоскелета и променама облика и покретљивости ћелија. На пример, ендотелне ћелије преусмеравају свој цитоскелет у смеру протока када су изложене смицајним напонима.

Узајамни однос између механике ћелије и физичких својстава њеног окружења пресудан је током миграције ћелије. Током ширења и миграције, адхезија ћелија за подлогу покреће преуређивање цитоскелета како би се покренула протрузија мембране и ширење ћелија. Да би мигрирале, ћелије такође користе места адхезије која се налазе на предњем делу ћелије као кортикална сидра за полимеризациону мрежу актина која се потискује против плазматске мембране и повлачи за подлогу док гура ћелијско тело напред. Како ћелија ствара нове адхезије на ивици ћелије, она мора да тестира механичку

отпорност ECM-а да би створила одговарајућу количину силе за оптимизацију миграције. [10]

1.2.1 Покретљивост индивидуалних ћелија

Пионирски рад на овом пољу потиче од изучавања у раду [7] када је откривено да се покретљивост појединих ендотелних ћелија може добро апроксимирати као упорно насумично ходање. Кратки временски период ћелије се континуирано крећу дуж праве линије. Насупрот томе, кретање је случајно (дифузивно) ако се истражује током довољно дугог временског оквира.

Брзина појединачних ћелија варира у зависности од услова, културе ћелија и присутних фактора раста, а утврђено је да је у опсегу од 10 - 50 μм. Ин виво, покретне ћелије ендокарда и ендотела, такође се крећу средњом брзином од приближно 20 μм / h. Време трајања, временска скала која раздваја линеарни и дифузни режим трајног случајног хода, регистрована је у распону од 0,6–3 сата у ћелијским културама.

1.2.2 Колективна миграција ћелија

Ендотелне ћелије такође показују способност координисане (колективне) миграције у култури. У одсуству усмереног ширења целог једног слоја, ове ћелије показују глобално неусмерено, али локално међузависно понашање струјања. Статистичка карактеризација кретања спонтаног струјања унутар једнослојних ендотелних слојева открила је да се ћелије крећу локално анизотропним, широким 50–100 μм и дугим 200–300 μм струјама, које се формирају и нестају на случајним положајима.

2. BPAE ћелије

Како је симулација вршена у наставку овог рада базирана на параметрима праћења миграције и пролиферације плућне артерије код говеда (BPAE), потребно је говорити и о карактеристика и понашању поменутих ћелија, као и зашто се у истраживањима проучава баш ова врста ћелија. Пролиферација ендотелних ћелија је важна за многе биомедицинске процесе попут ангиогенезе и санације оштећене васкулатуре.

Један од многих али значајнијих проблема јесте контрола раста ендотелних ћелија током зарастања ране након васкуларних повреда изазваних током поступака као што је ангиопластика. BPAE ћелије су такође позитивне на ензим који претвара ангиотензин (ACE – angiotensin converting enzyme), ензим који је задужен за одржавање притиска крви и запремине. Због ове чињенице, BPAE ћелије се често користе у истраживањима хипертензије, као и у студијама атеросклерозе и болести коронарних артерија.

Мотилитет и пролиферацију регулише неколико фактора раста полипептида. Ендотелне ћелије синтетишу основни фактор раста фибробласта (bFGF) и факторе раста изведене из тромбоцита (PDFG) ин витро, док неометане културе (тј. Културе које нису рањене механичким путем) ослобађају само PDFG. Трансформишући фактор раста β

(TGFβ) блокира деловање bFGF и инхибира кретање и митозу ендотелних ћелија артерије говеда код рањених слојева. Инхибиција је неутрализована после 24h стимулативним ефектом bFGF који су ослободиле ендотелне ћелије. Такође је откривено да је bFGF регулисао базални ниво синтезе протеазног активатора плазминогена (PA) и базални ниво синтезе ДНК, резултат који подржава ранија сазнања о моћним митогеним ефектима bFGF[11].

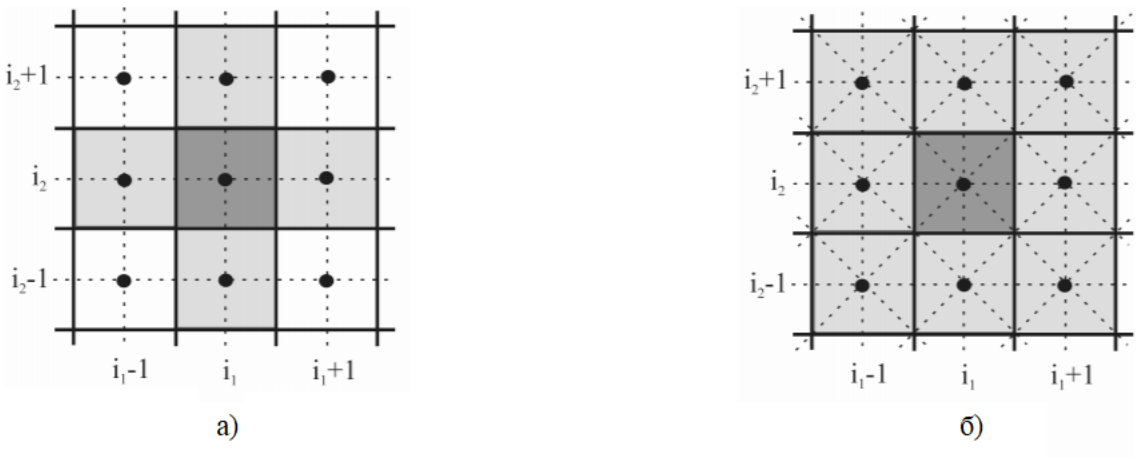
Све ове студије изнедриле су податке о покретљивости ћелија и њиховој међусобној интеракцији. Међутим за већину система нису познати фактори који су пресудни при моделирању понашања ћелија јер не постоји довољно квантитативних података који би омогућили израчунавање ових параметара кретања ћелије. Подаци показују да су мотилитет ћелија и њихов међусобни контакт пресудни фактори у одређивању пролиферације ћелија, према томе то су основни фактори које треба узети у обзир при моделирању било каквог раста ендотелних ћелија.

3. Ћелијски аутомати

Ћелијски аутомати су дискретни, апстракни рачунарски системи који се користе за симулирање различитих природних феномена који могу бити било линеарни, било нелинеарни. Ћелијски аутомати се састоје од коначног скупа ћелија дискретно распоређених у једну или више димензија (низ, матрица, …). Ћелије поседују коначан скуп правила помоћу које јој омогућавају да реагују са околином или да у зависности од околине мењају своје унутрашње стање. Ћелијски аутомати су временски дискретни, што значи да се стања ћелија и интеракција са околином врши у дискретним временским корацима, а не у било ком временском тренутку. У сваком временском кораку, једна ћелија може да пређе у само једно од својих стања, и у сваком временском кораку, свака ћелија синхронизовано врши прорачун у које стање да пређе.

Апстрактно својство ћелијских аутомата се односи на то да се ћелијски аутомати могу математички објаснити, и затим могу бити имплементирани на физичким структурама, попут рачунара. Понашање ћелијских аутомата се разликује од понашања Тјурингових машина (рачунара), али уз добро дефинисање правила, ћелијски аутомати могу да симулирају универзалну Тјурингову машину [12]. Као што је наведено изнад, ћелије мењају стања на основу пређашњег стања, и на основу околних ћелија. Код дводимензионих ћелијских аутомата, околина се најчешће посматра на два начина, и то као Мурова околина и Фон Нојманова околина.

Фон Нојманова околина узима у обзир четири суседне ћелије, и то једну северно, једну источно, једну јужно и једну западно од посматране ћелије.Мурова околина додаје још четири ћелије у суседство и то југоисток, југозапад, северозапад и североисток (Слика 3) [13].



Слика 3 Фон Нојманова (а) и Мурова околина (б) [13]

Примери ћелијских аутомата

1. Једнодимензиони ћелијски аутомат

У овом поглављу биће описан једноставни ћелијски аутомат, где су ћелије дискретно распоређене по једној димензији (у низу). Ћелије могу бити у једном од два стања:

1. Стање 0 – Ћелија је црне боје
2. Стање 1 – Ћелија је беле боје

Функција преноса стања је описана једначином (1):

$$a_i^{(t+1)} = a_{i-1}^{(t)} + a_{i+1}^{(t)} mod 2 \quad (1)$$

где је:

t+1 – следећи временски корак

t – тренутни временски корак

a_i – ћелија на *i*-тој позицији у низу

Према горе наведеној функцији, ћелија има вредност остатка при дељењу збира левог и десног суседа са два. Из таблице истинитости (1), може се закључити да је вредност ћелије једнака операцији искључиве дисјункције над левим и десним суседом. У почетном временском тренутку, све ћелије имају стање 0, осим ћелије у средини низа, која има стање 1 [14].

Табела 1: Таблица истинитости фунцкије преноса (1)

$a_i^{(t+1)}$	$a_{i-1}^{(t)}$	$a_{i+1}^{(t)}$
0	0	0
1	0	1
1	1	0
0	1	1

Алгоритам за ову једноставну ћелијску аутомату се може описати следећим корацима:

1. Иницијализација низа величине *n*
2. Попуњавање низа са почетним вредностима (сваки члан низа је 0 на почетку, осим члана $a_{n/2}$, који има вредност 1)
3. Исцртавање низа
4. Ископирати низ у нови, привремени низ, који означава стање система у следећем временском кораку
4. За сваки члан оригиналног низа a_i, где *i* припада скупу {*1,n-1*} извршити операцију $a_{i-1} \oplus a_{i+1}$, и резултат уписати на *i*-то место новог низа
5. Заменити стари низ са новим низом

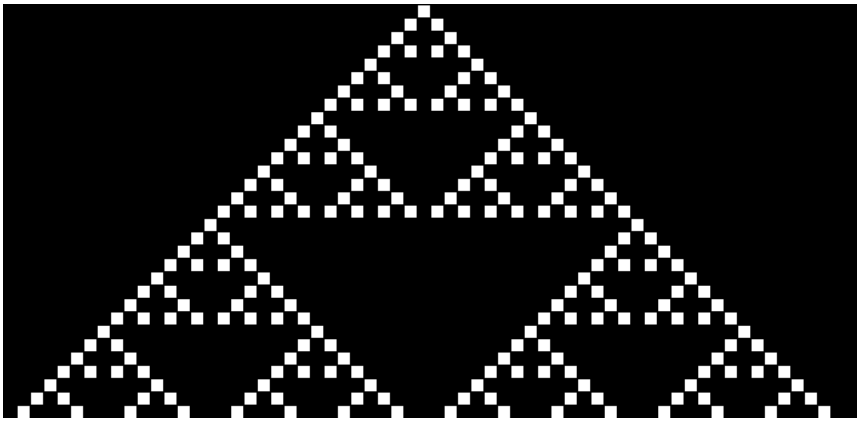
6. Враћање на корак 3.

У табели 2, могуће је видети резултате извршавања ћелијског аутомоата у прва три корака, за величину низа 5:

Табела 1: Прва три корака симулације

члан/корак	0	1	2	3	4
k=0	0	0	1	0	0
k=1	0	1	0	1	0
k=2	1	0	0	0	1

Програм описан алгоритмом изнад писан у *Processing Java* окружењу је дат у прилогу 1, и његовим извршавањем за првих тридесет корака, добија се конфигурација приказана на слици 2. Са слике 4, види се да ћелије граде фрактални шаблон, познатији као Троугао Сјерпињског где је сваки део шаблона сличан целом шаблону.



Слика 4 Симулација једнодимензионог ћелијског аутомата у првих 30 временских корака

2. Дводимензиони ћелијски аутомат

Један од најпознатијих ћелијских аутомата је „Игра Живота“ Џона Конвеја. Овај дводимензиони ћелијски аутомат је први пут описан седамдесетих година двадесетог века, састоји се од матрице ћелија са једноставним скупом правила:

1. Смрт ћелије: Ако је ћелија жива (стање 1), прећи ће у стање 0 (смрт) у следећим случајевима:

- Пренасељеност – ћелија има четири или више живих ћелија у својој околини

- Усамљеност – ћелија има мање од две живе ћелије у својој околини

2. Рођење ћелије: Ако је ћелија у стању 0 (смрт), ћелија ће проћи у стање 1 (живот) ако и само ако има три живе ћелије у својој околини

3. Стаза:

- Ако је ћелија жива и у њеној околини се налазе две или три живе ћелије, она ће остати у животу

- Ако је ћелија мртва, она остаје мртва ако нема тачно три живе ћелије у својој околини

Овај ћелијски аутомат, који има неколико једноставних правила је способан да произведе веома сложене облике, а као неки од облика, издвајају се три категорије, приказане на слици 5 [15]:

1. Објекти који су непокретни
    - Блок (а)
    - Кошница (б)
    - Хлеб (в)
    - Брод (г)

2. Објекти који осцилирају између два стања:
    - Трептач (д)
    - Жаба (ђ)
    - Светионик (е)

3. Објекти чија следећа генерација задржава облик, али мења позицију, одајући утисак кретања по матрици:
    - Једрилица (ж)
    - Свемирски брод (з)

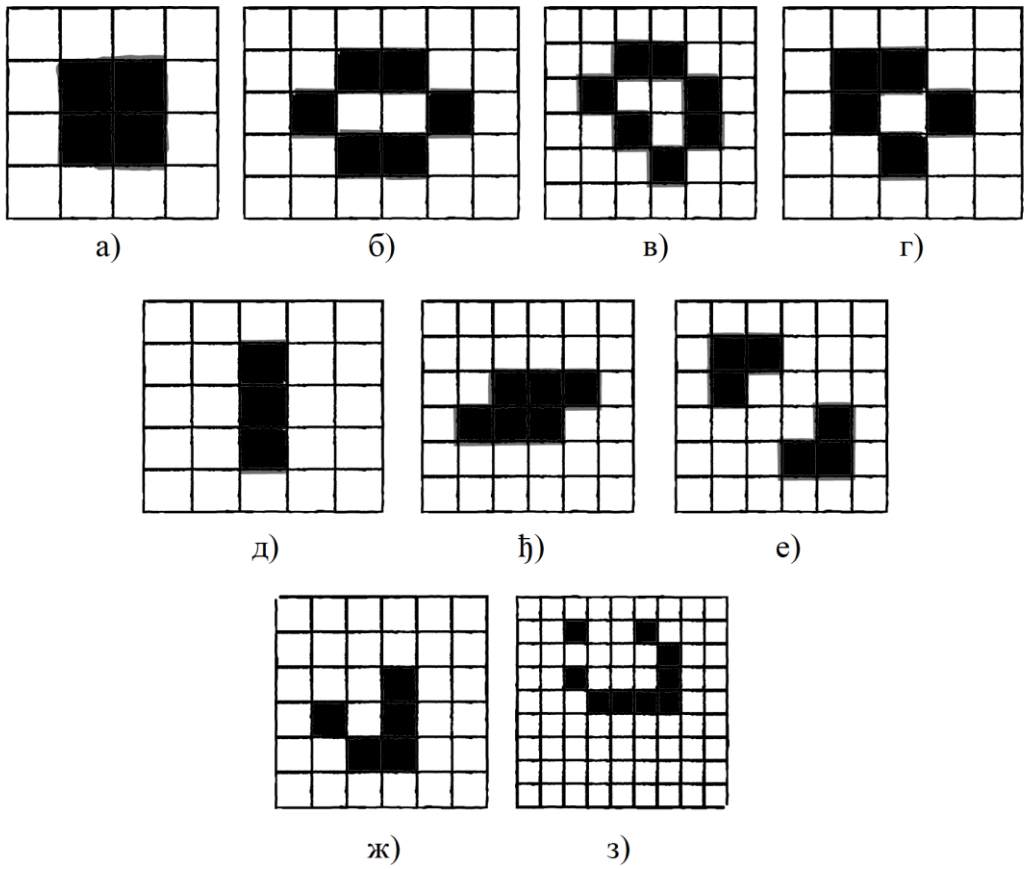
Алгоритам за ову врсту ћелијског аутомата је описан следећим корацима:

1. Иницијализација матрице димензије mxn

2. Попуњавање матрице иницијалним вредностима (могуће је насумично поставити живе ћелије)

3. Исцртавање матрице

4. Копирање матрице која садржи тренутно стање у нову, привремену матрицу



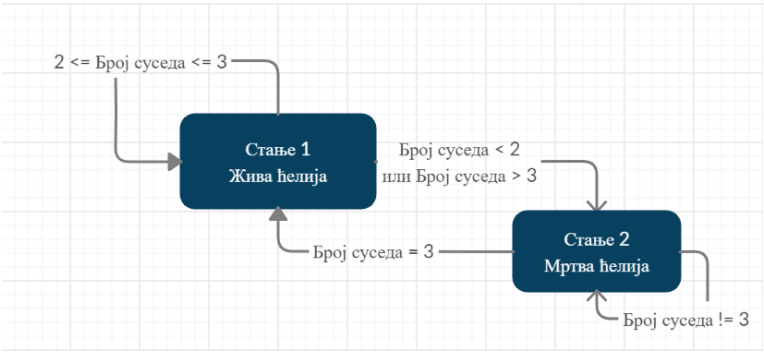
Слика 5 Чести шаблони који се појављују у Игри Живота [15]

5. Итерација кроз стару матрицу, за сваку ћелију примењивање правила дефинисаних на почетку поглавља (провера околине ћелија, и одлучивање о новом стању), и резултат калкулације сачувати у привремену матрицу

6. Копирање привремене матрице у стару матрицу

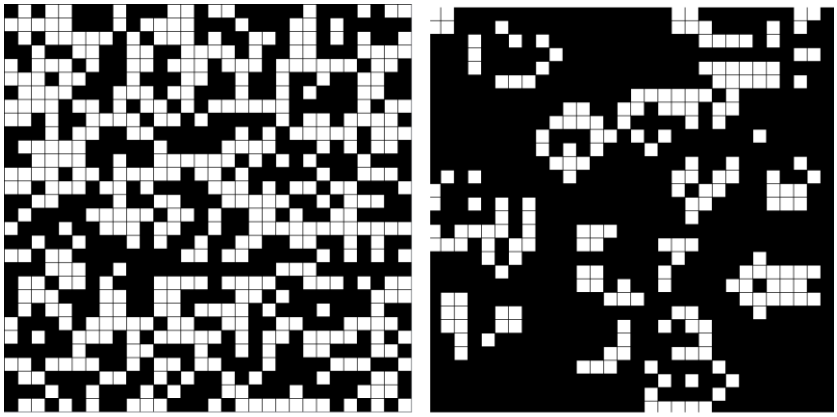
7. Понављање трећег корака

По правилима наведеним изнад, може се видети да се свака ћелија понаша као машина коначних стања, те је дијаграм транзиције стања приказан на слици 6.



Слика 6 Дијаграм стања једне ћелије у Игри Живота

Резултат извршавања Игре Живота, чији је листинг дат у прилогу 2, за временске кораке 1 и 10, може се видети на слици 7.



Слика 7 Резултат извршавања Игре Живота у временским корацима 1 (а) и 10 (б)

Поновним покретањем програма, и насумичним распоредом живих ћелија у почетку симулације, резултати се драстично разликују, за разлику од примера једнодимензионог аутомата, где се концизно добија фрактални троугао. Из тога произилази класификација ћелијских аутомата на четири класе [16]:

1. Ћелијски аутомати који после одређеног времена дођу увек у хомогено стање у целом простору

2. Ћелијски аутомати који испољавају једноставне понављајуће шаблоне, или осцилују између између два или више шаблона после одређеног времена

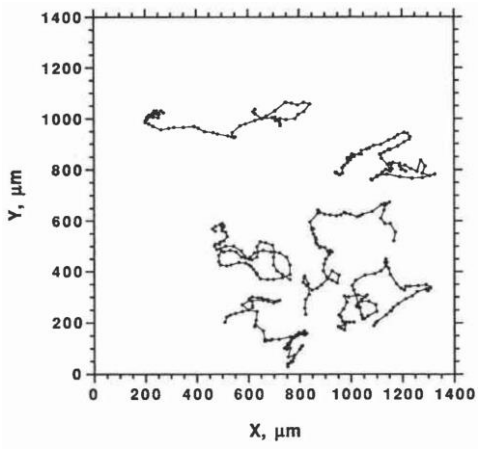
3. Ћелијски аутомати који испољавају хаотично апериодно понашање

4. Ћелијски аутомати који поседују комплексне локализоване структуре које се могу наћи на насумичним локацијама у простору

Уз помоћ дате класификације, закључује се да једнодимензиони пример спада у класу 2, а дводимензиони у класу 4.

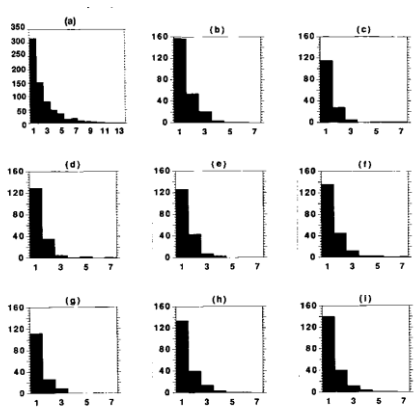
5. Ћелијски аутомат за моделирање миграционих ћелија са контактном инхибицијом

Пролиферацију миграторних ћелија описаних у претходним поглављима је могуће симулирати уз помоћ ћелијског аутомата описаног у поглављу 3. Миграторне ћелије испољавају перзистентну насумичну шетњу, где је анализа кретања вршена у радовима [11], [17]. На слици 8, приказана је трајекторија насумичне шетње миграторних ћелија током 36 сати.



Слика 8 Трајекторија миграторних ћелија током 36 сати [17]

Са слике 8 се види да ћелија одржава кретање у једном правцу одређено време, пре него што промени правац. Мера проведеног времена у једном стању се зове време чекања. Време чекања се разликује између стања и ћелија, и емпиријски добијена дистрибуција у [17] је приказана на слици 9.



Слика 9 Хистограм времена чекања за различите правце ћелије (а) стационирана ћелија, (б-и) смерови ћелије [17]

Хоризонтална оса показује временски корак у ком ћелија мења правац, а вертикална оса приказује колико пута је ћелија добила одређено време чекања. Из слике 7 и [17] може се увидети експоненцијална времена чекања, где у највећем броју случајева, ћелија мења правац у првом временском кораку. Други важан фактор у симулирању пролиферације миграторних ћелија са контаткном инхибицијом јесте време деобе ћелија. У [1], експериментално је утврђено да се 64% ћелија подели између 12 и 18 сати, 32% између 18 и 24 часа, а преосталих 4% између 24 и 30 сати.

Место где ће се наћи нова ћелија се одређује помоћу вероватноћа раста, и вероватноћа раста износи 1 за Фон Нојманову околину, а за остатак околине која чини Мурову околину може се бирати између 0 и 1 [18]. Трећи параметар представља смер кретања ћелија кроз време. Према [1], однос вероватноћа за промену смера за 180^o, 135^o, 90^o, и 45^o је 1:0.77:2.04:9.73, респективно [1]. Ћелије могу доћи и у стационарно стање, кога одликује одржавање претходног стања, било кретања у неком смеру, било ћелијске стазе. Анализом уз помоћ Марковљевог ланца, закључено је да ако се ћелија налази у мировању, 80% је вероватноћа да ће она остати у мировању. У случају да се ћелија креће, вероватноћа да настави да се креће истом путањом је око 2% [11].

Због природе проблема који се описује, који захтева кретање ћелија, као и деобу, ћелије у овом ћелијском аутомату морају да чувају више од једног стања, и то:

- Смер у коме се ћелија креће
- Време дељења ћелије
- Време одржавања кретања у једном смеру
- Тренутну брзину ћелије

Будући да се ћелије крећу у дискретним корацима, и прелазе фиксну дистанцу између два корака, није потребно да се чува тренутна брзина ћелије. Просечна брзина ћелије се може варирати дефинисањем временског корака, као и вероватноћа за промену смера ћелија. Узевши све у обзир, стање ћелије може да се дефинише четвороцифреним целим бројем *klmn*, где је:

- *k*=[0,8] индекс смера у коме се ћелија креће, 0 означава мировање ћелије, док 1-8 означавају индекс у негативном математичком смеру, где је 1 смер ка северу (видети слику 1б)
- *l*=[0,9] време чекања. Када бројач дође на 0, долази до процедуре за деобу ћелија. Бројач се иницијализује према горенаведеној експоненцијалној дистрибуцији приказаној на слици 7.
- *mn*=[0,99] бројач деобе ћелија. Када бројач дође на нулу, наступа деоба ћелија. Бројач се иницијализује према вероватноћама наведеним у тексту изнад.

Ако ћелија, на пример, има стање 1427, то значи да се креће у правцу севера још 4 временска корака, а да ће деоба наступити за 27 временских корака. Као основа за

симулацију, узима се да временски корак траје пола сата. Узима се да је величина ћелије 28 μm, и да ћелија може да се помери за максимално једно место током једног временског корака. То значи да просечна брзина ћелије директно зависи од величине временског корака, и за временски корак од пола сата, брзина кретања ћелија у правцу север, исток, југ, запад може бити 56 μm/h, док у правцима североисток, северозапад, југоисток и југозапад (дијагонално) може бити 79 μm/h. За просечну брзину, узима се просек брзина целе колоније ћелија [1].

5.1 Алгоритам ћелијског аутомата за симулацију миграторних ћелија са контактном инхибицијом

У овом поглављу дат је алгоритам ћелијског аутомата за симулацију миграторних ћелија са контактном инхибицијом без освртања на имплементационе детаље, који су обрађени у следећем поглављу. Приликом иницијализације симулације, задају се почетне локације ћелија, и то насумично по матрици у задатом броју (густина насељености), или испуњавањем центра матрице одређеним бројем ћелија. Приликом иницијализације ћелија, свака ћелија добија насумично одабран четвороцифрен број који означава стање описано у тексту изнад, при томе поштујући горе поменуте дистрибуције за различите бројаче (деоба, кретање, смер). По завршетку корака иницијализације, следи петља која се понавља у сваком временском тренутку, са следећим редоследом извршавања:

1. Направити нову, привремену, празну матрицу стања
2. Насумичан одабир једног члана из старе матрице стања
3. Ако члан садржи ћелију, и ако је време за деобу ћелије, извршити рутину за деобу и прећи на корак 6
4. Ако члан садржи ћелију, и ако је време за промену смера кретања, извршити рутину за промену смера и прећи на корак 6
5. У супротном, покушати померај ћелије у смеру који је забележен у ћелији, и ако је место слободно у новој матрици, уписати стање ћелије на то место, и смањити бројаче за деобу и за промену смера. Ако је место у новој матрици заузето, у новој матрици поставити ћелију на исто место на коме је била у старом, и смањити бројач за промену смера на нулу (промена смера се онда извршава у следећем временском кораку)
6. Насумични одабир једног од преосталих чланова из старе матрице, и враћање на корак 3, све док се сваки члан не прође
7. Освежити стару матрицу стања са вредностима из нове матрице
8. Исцртавање матрице стања

5.1.1 Рутина за промену смера:

1. Скенирање Мурове околине ћелије и провера да ли постоји слободно место.

2. Ако не постоји слободно место, ћелија остаје на истом месту.

3. Ако постоји слободно место, постоји вероватноћа да ће ћелија остати у месту, или наставити кретање у истом смеру (ако је место за кретање у том смеру слободно) по дистрибуцији која је наведена у поглављу изнад.

4. Ако ћелија не долази у стационарно стање, онда променити смер ћелије по вероватноћама за промену смера датим у поглављу изнад.

5. У новој матрици стања, означити место на коме ће се ћелија налазити у следећем временком кораку, иницијализовати ново време чекања по горе наведеној дистрибуцији

6. Смањити бројач деобе за један.

5.1.2 Рутина за деобу ћелије:

1. Скенирање Мурове околине ћелије и провера да ли постоји слободно место.

2. Ако слободно место не постоји, ћелија се неће поделити.

3. Ако постоје слободна места у околини, насумично одабрати једно место по вероватноћама раста

4. Означити место нове ћелије у новој матрици стања, иницијализовати нову ћелију, иницијализовати бројач деобе на старој ћелији на нову вредност. [1]

5.2 Имплементациони детаљи ћелијског аутомата

У поглављу изнад, показано је да се итерација кроз све елементе матрице стања не врши редоследно, већ насумично. Разлог томе је превенција привржености кретања колоније ћелија у смеру по којем се итерира кроз матрицу, јер због провере околине, сваки елемент би истим редоследом био уписан у нову матрицу, што би стварало различите рачунске артифакте због скенирања околине. Насумичним одабиром се тај проблем превазилази, али због спорости алгоритма за генерисање псеудонасумичних бројева, позивање тог алгоритма онолико пута колико има чланова матрице би утицало на перформансе целе симулације. Решење тог проблема се огледа у прављењу два низа, од којих први садржи индексе редова, а други индексе колона матрице стања. Потом, у сваком временском кораку, довољно је само "промешати" индексе, и онда приступати члановима матрице помоћу тих индекса итерацијом кроз низове. Тиме се временска комплексност насумичног одабира смањује са mn на $n+m$, односно за квадратну матрицу

са n^2 на $2n$ [8]. У листингу 1 је дато почетно постављање низа и процедура за мешање чланова.

***Листинг 1**: Иницијална поставка низова индекса и мешање чланова*

```
int[][] indices;

 void setup(){
   //inicijalizacija indeksa, u prvom redu se nalaze indeksi redova
   // u drugom indeksi kolona
   indices = new int[2][100];
   for (int i = 0; i < 2; i++) {
     for (int j = 0; j < 100; j ++) {
      //popunjavanje indeksa redosledno
       indices[i][j]=j;
     }
     //mešanje indeksa
     shuffle(indices[i]);
   }
 }

 void shuffle(int[] array) {
   //prolazi se kroz svaki član niza
   for (int i = 0; i < array.length; i++) {
     //bira se nasumični član, sa kojim će trenutni član da zameni mesto
     int index = random.nextInt(i + 1);
     //ako odabrani član nije isti koji je trenutno na redu
     if (index != i)
     {
       //zameniti im pozicije
       array[index] ^= array[i];
       array[i] ^= array[index];
       array[index] ^= array[i];
     }
   }
 }
```

Како је утврђено у поглављу изнад, времена за дељење ћелија су подељене у три категорије, са одређеним вероватноћама за дељење између 12 и 18 часова, између 18 и 24, и између 24 и 30. Времена дељења ће се чувати у низу са 100 чланова, где један члан представља један проценат вероватноће. Будући да је вероватноћа да се ћелија подели између дванаестог и осамнаестог сата 64%, првих 64 чланова низа се попуњавају са насумичним бројевима између 12 и 18. Пошто је временски корак симулације пола сата, а не цео сат, онда се број множи са два, тако да се првих 64 чланова пуне насумичним вредностима између 24 и 36. Процедура се понавља и за остале две категорије. У листингу 2 је дата метода која попуњава низ који ће служити за узимање вредности бројача деобе. При сваком иницијализовању бројача деобе, узима се насумични члан овог низа.

***Листинг 2**: Попуњавање низа са вредностима времена деобе ћелије*

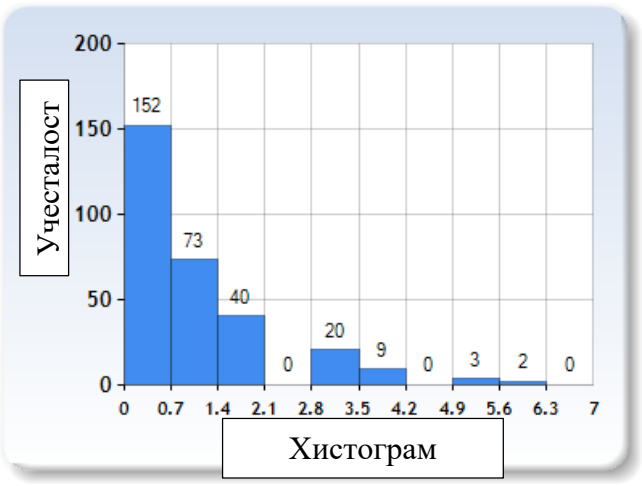
```
void populateDivisionDistributionArray(){
  for (int i = 0; i < 100; i++){
    if (i < 64){
      divisionDistribution[i] = 24 + int(random(12));
    } else if (i < 96){
      divisionDistribution[i] = 36 + int(random(12));
    } else {
      divisionDistribution[i] = 48 + int(random(12));
    }
  }
}
```

Из хистограма дистрибуције времена чекања за промену смера (слика 7), може се закључити да је у питању инверзна експоненцијална дистрибуција. Будући да вредности ове дистрибуције припадају скупу [0, ∞), потребно је скратити опсег на [0, 1). Функција је дата формулом 2 [19].

$$x = \frac{-ln(1-(1-e^{-\lambda})*U)}{\lambda} \tag{2}$$

где је U насумичан број [0, 1), а λ – параметар раста криве.

Варијацијом параметра раста криве и позивањем методе за генерисање насумичног броја овом врстом дистрибуције, те исцртавањем хистограма за резултате, долази се до закључка да се за параметар раста $\lambda=6$ добија дистрибуција који највише одговара дистрибуцији на слици 9 (Слика 10).



Слика 10 Хистограм дистрибуције насумичних бројева позивањем методе за инверзну експоненцијалну дистрибуцију

У листингу 3 је дата метода за добијање насумичног броја који прати инверзну експоненцијалну дистрибуцију, у опсегу [0, 1).

***Листинг 3**. Метода за генерисање насумичног броја који потпада под инверзну експоненцијалну дистрибуцију*

```
float logRandom(){
  return -log(1 - (1 - exp(-6)) * random(1)) / 6;
}
```

Приликом настанка ћелије, она има једнаку вероватноћу да се креће у било ком од осам праваца, или да стоји у месту. Узевши у обзир горе наведене функције, могуће је направити методу која креира и враћа стање новонастале ћелије (листинг 4).

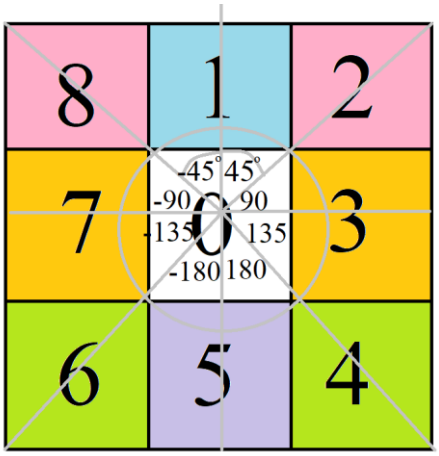
***Листинг 4**. Метода за креирање нове ћелије и враћање њеног стања*

```
int createCell(){
  //odabir pravca
  int direction = int(random(9)) * 1000;
  //odabir vremena kretanja u tom pravcu, poštujući inverznu eksponencijalnu
distribuciju
  int persistenceCounter = int(logRandom() * 10) * 100;
  //odabir vremena do deobe, poštujući distribuciju vremena
  int divisionCounter = divisionDistribution[int(random(100))];
  return direction+persistenceCounter+divisionCounter;

}
```

Рутина за промену смера је нешто сложенија, и захтева више операција. На почетку, потребно је у низ ставити све слободне правце у околини ћелије. Потом, потребно је проверити да ли ће ћелија остати у месту, или се кретати старим правцем (ако је слободан) посредством насумичног избора. Ако се тај услов не испуни, и ћелија треба да заузме нови правац, потребно је извршити рутину налик оној за избор времена деобе. Будући да су различите вероватноће за различите углове промене смера, и износе за 180^o, 135^o, 90^o, и 45^o 1:0.77:2.04:9.73, потребно је проверити који су углови доступни, и тај смер ставити онолико пута у листу колика је његова вероватноћа да се деси. Пошто ћелија може променити смер за неки фиксни угао, тај угао је симетричан око тренутног смера, у смислу да је иста вероватноћа да ћелија крене на лево за тај угао, као и на десно. Како се не би повећала вероватноћа за дупло да ћелија промени смер у један угао, ако је тај угао слободан и лево и десно, већ да оба угла деле једнаку вероватноћу, потребно је проверити да ли су доступна оба смера, и ако јесу, онда укупну вероватноћу за тај смер поделити на та два угла, а у супротном укупну вероватноћу дати једном слободном. Будући да се ради о Муровој околини, где је север угао 1, угао од 45 степени у односу на север би био на 2 и 8 (слика 11). Угао од 45 степени је +/- једно поље у односу на тренутни смер, 90 степени +/- 2, и тд. Пошто је последњи, осми, члан Мурове околине у додиру са првим чланом, рутина за проверу угла би се сводила на сабирање тренутног угла са инкрементом новог угла, узимати остатак при дељењу са 8, и одузимање од тренутног и

узимање остатка при дељењу са 8. Ако се за резулатат добије 0, онда променити резултат у 8, јер је остатак при дељењу са 8 једнак нули. Овај резултат даје које поље из мурове околине заузима дат угао, и онда се лако може проверити да ли је поље доступно (добијено у прошлом кораку), и ако јесте, ставити то поље у низ онолико пута колика му је вероватноћа. Описане рутине су дате у листингу 5.



Слика 11 Углови поља у околини за смер север (1)

Листинг 5. *Рутине за промену смера ћелије*

```
void compute() {
.
.
.
        if (directionCount(currentCell) == 0) {
          int dir = direction(currentCell);
          List<Integer> freeDirections = new ArrayList();
          for (int t = 1; t < 9; t++) {
            if (freeDirection(t, indices[0][i], indices[1][j], nextState)) {
              freeDirections.add(t);
            }
          }
          dir = newDirection(dir, freeDirections);
        }
.
.
.
}
int newDirection(int oldDirection, List<Integer> possibleDirections){
   directionDistribution.clear();
  //ako ćelija stoji u mestu, 80 posto je šansa da ostane u mestu
  //ako ne stoji u mestu, šansa da nastavi kretanje u prethodnom smeru je 5%
  if (oldDirection == 0){
    if ((int)random(10)>1){
      return 0;
    } else {
      if (possibleDirections.isEmpty()){
        return 0;
      } else {
        return possibleDirections.get((int) random(possibleDirections.size()));
      }
```

```
  } else if (possibleDirections.contains(oldDirection)){
     directionDistribution.add(oldDirection);
   } else if (possibleDirections.isEmpty()){
     return 0;
   }
```

```
  //+- 45 stepeni
  addDirections(getDirections(oldDirection,1, possibleDirections), 97);
  //+- 90 stepeni
  addDirections(getDirections(oldDirection,2,possibleDirections), 20);
  //+-135 stepeni
  addDirections(getDirections(oldDirection,3,possibleDirections), 8);
  //+-180 stepeni
  addDirections(getDirections(oldDirection,4,possibleDirections), 10);
  return directionDistribution.get((int)random(directionDistribution.size()));
}
List<Integer> directions = new ArrayList();
List<Integer> getDirections(int oldDirection, int newDirection, List<Integer>
possibleDirections){

  directions.clear();
  int direction = abs(8 + oldDirection - newDirection) % 8;

  if (direction == 0){
    direction = 8;
  }
  if (possibleDirections.contains(direction)){
    directions.add(direction);
  }
  direction = abs(8 + oldDirection + newDirection) % 8;
  if (direction == 0){
    direction = 8;
  }
  if (possibleDirections.contains(direction)){
    directions.add(direction);
  }

  return directions;
}
void addDirections(List<Integer> directions, int count){
  if (directions.isEmpty()){
    return;
  }
  for (int i = 0; i < count; i++){
    if (i < count/directions.size()){
      directionDistribution.add(directions.get(0));
    } else {
      directionDistribution.add(directions.get(1));
    }
  }
}
```

```
boolean freeDirection(int dir, int i, int j, int[][] nextState) {

  switch(dir) {
  case 0:
    return true;
  case 1:
    if (i+1 < 100 && nextState[i+1][j] == -1 ) {
      return true;
    }
    break;
  case 2:
    if (i+1 < 100 && j-1 >= 0 && nextState[i+1][j-1] == -1) {
      return true;
    }
    break;
  case 3:
    if (j-1 >= 0  && nextState[i][j-1] == -1) {
      return true;
    }
    break;
  case 4:
    if (j-1 >= 0 && i - 1 >= 0 && nextState[i-1][j-1] == -1) {
      return true;
    }
    break;
  case 5:
    if (i - 1 >= 0 && nextState[i-1][j] == -1) {
      return true;
    }
    break;
  case 6:
    if (j + 1 < 100 && i - 1 >= 0 && nextState[i-1][j+1] == -1) {
      return true;
    }
    break;
  case 7:
    if (j + 1 < 100 && nextState[i][j+1] == -1) {
      return true;
    }
    break;
  case 8:
    if (j + 1 < 100 && i + 1 < 100 && nextState[i+1][j+1] == -1) {
      return true;
    }
    break;
  }
  return false;
}
```

Рутина за деобу ћелија је нешто једноставнија, скенирају се све ћелије у околини. Када се наиђе на празно поље, њене координате се чувају у два низа, по један за обе осе. Претрага се наставља, и ако се наиђе на још неко празно поље, координате се додају у ова два низа. Операција се понавља док се цело окружење ћелије не скенира. Прави се нова ћелија на насумичном празном пољу, а на тренутном пољу се тренутна ћелија реиницијализује. Из методе се враћа да ли је ћелија успешно подељена или не. Ако ћелија није подељена, наставља се са алгоритмом на остале делове петље, као што је промена смера и миграција, а ако јесте, прелази се директно на рачун за нову ћелију. Рутина је описана листингом 6.

Листинг 6. *Рутина за деобу ћелије*

```
List<Integer> freeCellX = new ArrayList();
List<Integer> freeCellY = new ArrayList();
boolean divide(int i, int j, int[][] nextState) {
  freeCellX.clear();
  freeCellY.clear();
  int newX = 0;
  int newY = 0;
  boolean found = false;
  for (int x = -1; x < 2; x ++) {
    for (int y = -1; y < 2; y ++) {
      if (i + x >= 0 && j + y >= 0  && i + x < 100 && j + y < 100 &&
nextState[i+x][j+y] == -1) {
        if (i+x == i && j+y == j) continue;
        freeCellX.add(i+x);
        freeCellY.add(j+y);
        if (found) {
          if (int(random(2)) == 0) {
            newX = i+x;
            newY = j+y;
          }
        } else {
          found = true;
          newX = i+x;
          newY = j+y;
        }
      }
    }
  }
  if (!freeCellX.isEmpty()) {
    int index = random.nextInt(freeCellX.size());
    nextState[freeCellX.get(index)][freeCellY.get(index)] =
createCell();
  }
  nextState[i][j] = createCell();
  return found;
}
```

Листинг 7 описује рутину за миграцију ћелије у задатом правцу. Прво се проверава да ли ћелија може да мигрира у том правцу, и ако може, ћелија мигрира, а ослобађа тренутно место. У супротном, поставља бројач времена чекања на нулу, да би у следећем циклусу променила правац. Такође, брзина ћелије се додаје у укупну брзину, која се на крају циклуса дели са бројем ћелија да би се добила просечна брзина. У следећем листингу,

неки од случајева су изостављени због понављања, док се комплетна процедура може видети у прилогу 3, са остатком програма.

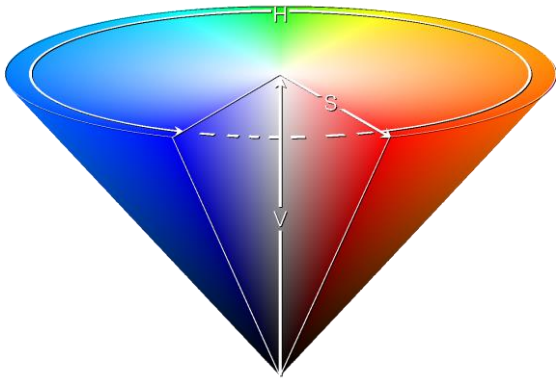
Листинг 7. Процедура за миграцију ћелије

```
void migrate(int currentCell, int i, int j, int[][] nextState) {
  int div = divisionCount(currentCell);
  int dirCnt = directionCount(currentCell);
  nextState[i][j] = -1;
  switch(direction(currentCell)) {
  case 0:
    break;
  case 1:
    if (i+1 < 100 && nextState[i+1][j] == -1) {
      i+=1;
      distance+=28;
    } else {
      dirCnt = 0;
    }
    break;
  case 2:
    if (i+1 < 100 && j-1 >= 0 && nextState[i+1][j-1] == -1) {
      i += 1;
      j -= 1;
      distance+=1.41*28;
    } else {
      dirCnt = 0;
    }
    break;

  case 7:
    if (j + 1 < 100 && nextState[i][j+1] == -1) {
      j += 1;
      distance+=28;
    } else {
      dirCnt = 0;
    }
    break;
  case 8:
    if (j + 1 < 100 && i + 1 < 100 && nextState[i+1][j+1] == -1) {
      j += 1;
      i += 1;
      distance+=1.41*28;
    } else {
      dirCnt = 0;
    }
    break;
  }
  div -= div > 0 ? 1 : 0;
  dirCnt -= dirCnt > 0 ? 1 : 0;
  nextState[i][j] = direction(currentCell) * 1000 + dirCnt * 100 + div;
}
```

Исцртавање ћелија се врши у *HSB(Hue, Saturation, Brightness)* простору боја (слика 12). Уместо удела примарних адитивних боја (црвена, зелена и плава), у овом простору боја координата боје се дефинише нијансом, засићеношћу и осветљењем боје. Овај простор боја је захвалнији за визуализацију неке величине јер функција боје по величини која се визуализује може бити линеарна у односну на ту величину, у смислу да је могуће мењати само једну координату (нијансу) за различите вредности мерене величине. На овај начин се визуализују подаци у различитим програмима за, на пример, визуализацију помераја добијених методом коначних елемената. Процедура за исцртавање ћелија пролази кроз целу матрицу стања, и ако у тренутном члану има ћелија она је исцртава по следећим правилима и дата је у листингу 8:

- Боја ћелије зависи од времена када ће доћи до деобе те ћелије
- Осветљење зависи од времена када ће доћи до промене смера ћелије



Слика 12 HSB простор боја. Угао на кружници (H) представља нијансу боје, растојање од центра кружнице (S) представља засићеност, а висина (V)

***Листинг 8**. Метода за исцртавање ћелија*

```
void drawCells() {
  background(0);
  for (int i = 0; i < 100; i++) {
    for (int j = 0; j < 100; j++) {
      if (playground[i][j] != -1) {
        if (playground[i][j] < 10000) {
          stroke(255-divisionCount(playground[i][j]), 100, 10 -
directionCount(playground[i][j]));
          fill(divisionCount(playground[i][j]), 100, 10 -
directionCount(playground[i][j]));
          rect(i*12, j*12, 12, 12);
        } else {
          stroke(100, 0, 100);
          fill(100, 0, 100);
        }
        rect(i*12, j*12, 12, 12);
      }
    }
  }
}
```

Почетни параметри симулације се постављају у методи *void setup()*, и они се тичу величине и распореда иницијалне колоније, као и временског корака *step* којим се ефективно мења брзина кретања ћелија, како је описано у корацима изнад. У симулацији су коришћене две поставке локације иницијалне популације ћелија, и то насумично по матрици, и концентрисана популација у центру матрице. Насумична поставка иницијалне популације се врши тако што се изабере број колико ће иницијално ћелија постојати, и онда се у петљи која се понавља онолико пута колики је тај број изабере насумична *X* и *Y* координата на матрици, и позива се процедура за генерисање ћелије на том месту. За иницијалну популацију концентрисану у центру матрице, користи се једначина круга (3), где се на свако поље које задовољава услов поставља нова ћелија позивањем процедуре за генерисање ћелије. Површина кружнице представља иницијалан број ћелије, а једини променљиви параметар је r^2. Дељењем жељеног броја ћелија са π, добија се колики параметар r^2 треба бити.

$$(x - x_c)^2 + (y - y_c)^2 < r^2 \tag{3}$$

У листингу 9, дата је конфигурација са иницијалном поставком од тридесет и две ћелије у концентрисане у центру, и 1200 ћелија насумично постављених.

Листинг 9. *Иницијализација популације*

```
void setup(){
.
.
.
  //postavljenje 32 celije u centar matrice
  for (int i = 0; i < 100; i ++) {
    for (int j = 0; j < 100; j ++) {
      if ((i - 49) * (i - 49) + (j - 49) * (j - 49) < 10) {
        playground[i][j] = createCell();
    }
  }
  //postavljanje 1200 celija na nasumicna mesta
  for (int i = 0; i < 1200; i++){
    playground[(int)random(100)][(int)random(100)] = createCell();
  }

.
.
.
}
```

6. Резултати и дискусија

Симулација миграторних ћелија помоћу ћелијског аутомата извршава се на матрици димензије 100 редова и 100 колона. Извршено је више симулација са варијацијом брзине ћелија, иницијалном величином популације и иницијалним распоредом ћелија. Све симулације су урађене брзином од 240 корака у секунди. Варијација брзине симулације не утиче на резултат симулације, већ само на графички приказ. Симулације су обављене на следећем софтверу и хардверу:

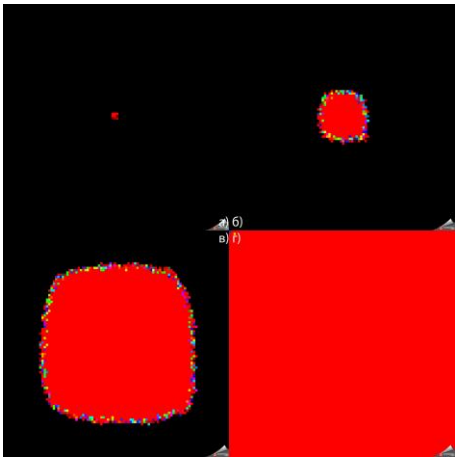
- Microsoft Windows 10 Professional
- Java SE 12
- Processing 3
- AMD Ryzen 3 2200g Quad Core 3.5 GHz
- 16 GB DDR4 RAM
- nVidia GeForce GTX1050 2 GB GDDR5 128 bit
- Kingston 480 GB SATA 3 SSD

На слици 13, приказани су резултати симулације за пролиферацију непокрених ћелија.

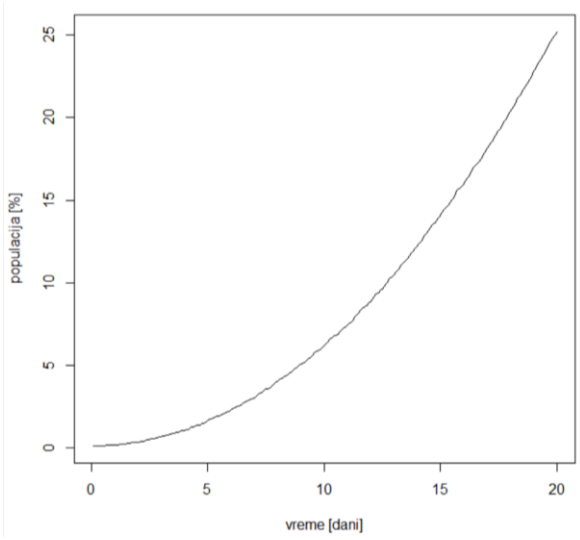
Параметри симулације су:

- Инцијална популација од 9 ћелија (0.09%)
- Ћелије концентрисане у средини матрице
- Временским кораком T = 0.5 h (*step = 0.5)*
- Трајање симулације 8 секунди

На слици 14, приказан је график раста популације у односу на време.



Слика 13 Симулација пролфиерације непокретних ћелија, почевши од горе лево у смеру казаљке а) Инцијална колонија б) Колонија после 5 дана в) колонија после 25 дана г) колонија после 40 дана



Слика 14 График раста популације за непокретне ћелије

Са слике 14, уочава се експоненцијални раст ћелија у првих 10 дана. У следећој симулацији, ћелије се крећу брзином од 3.5 микрометара по секунди. Параметри симулације су:

- Инцијална популација од 9 ћелија (0.09%)
- Ћелије концентрисане у средини матрице
- Временским кораком T = 8 h (*step = 8)*
- Трајање симулације 0.3 секунде

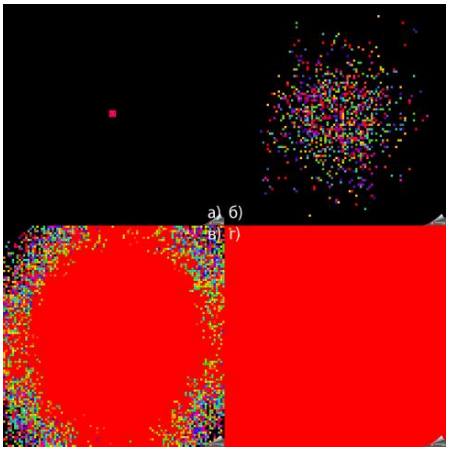
На слици 15 приказана је симулација у четири временска корака, док је на слици 16 приказан график функције раста ћелија у односу на време у данима. Поређењем графгика са слике 12 и слике 16, уочава се бржа пролиферација ћелија када су покретне у односну на непокретне ћелије. То се исто може видети и из графичког приказа симулација (сл 13., сл. 15), где је популација направила конфлуентан слој у случају непокретних ћелија за 40 дана, а покретне ћелије су то учиниле за нешто више од 25 дана.

Убрзањем симулације на 28 микрометара по секунди, популација испуњава 100% матрице за 10 дана и 20 сати. У овом случају, параметри симулације су:

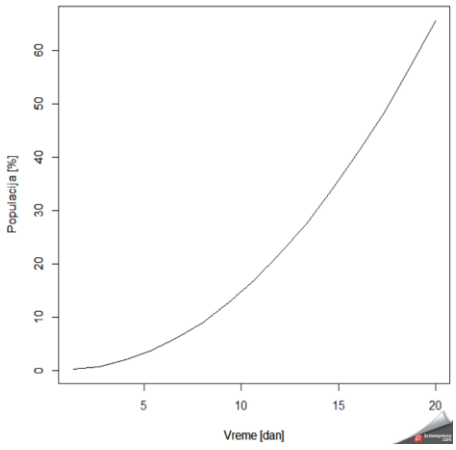
- Инцијална популација од 9 ћелија (0.09%)
- Ћелије концентрисане у средини матрице
- Временским кораком T = 1 h (*step = 1)*

- Трајање симулације 2.5 секунде

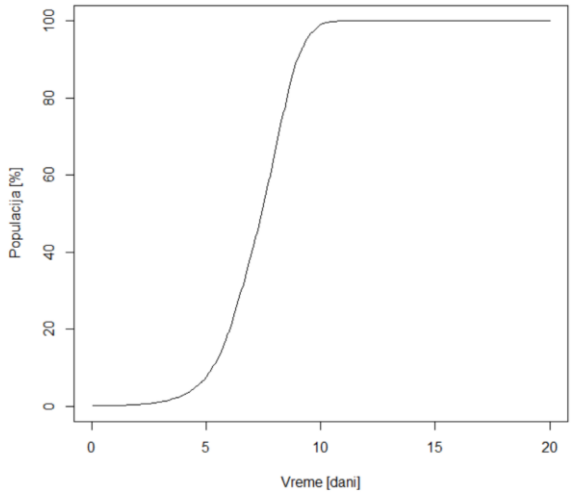
Са слике 17, види се да су ћелије знатно покретљивије, што доводи до деобе на насумичним местима, која су на почетку симулације неиспуњена, те се пролиферација одвија знатно брже. На слици 18, могуће је видети график раста популације при овој брзини миграције. Повећањем брзине за 100% на 56 микрометара по сату(слика 19 и 20), пролиферација се убрзава за нешто више од 10%, што имплицира нелинеарну зависност пролиферације од брзине кретања ћелија. У овом случају, временски корак је 0.5, а време извршења симулације је 1.8 секунди (9 дана).



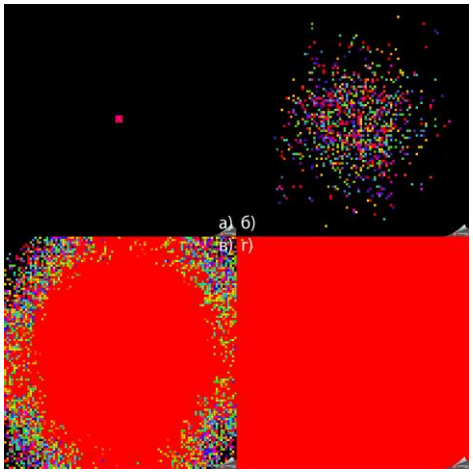
Слика 15 Симулација пролиферације ћелија са просечном брзином 3.5 микрометара по сату а) Иницијална колонија б) Колонија после 5 дана в) Колонија после 15 дана г)Колонија после 25 дана



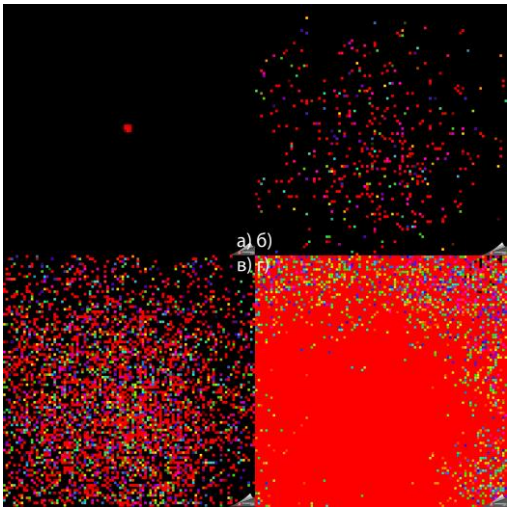
Слика 16 Функција раста ћелија у односу на време, просечна брзина 3.5 микрометара по сату



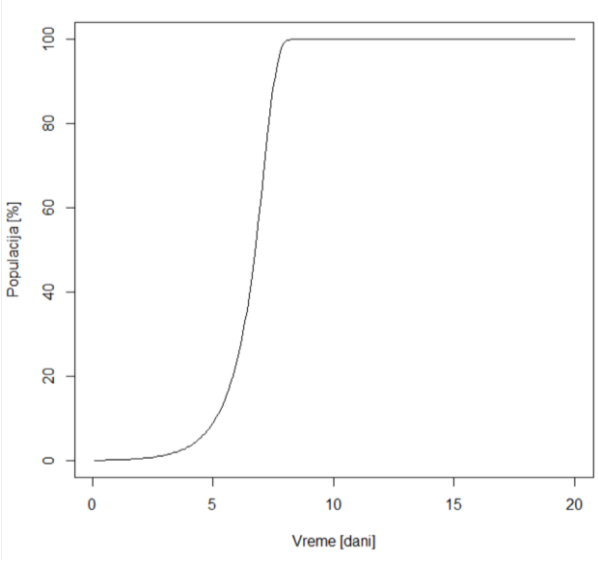
Слика 17 Функција раста колоније ћелија у зависности од времена, брзина 28 микрометара по сату



Слика 18 Симулација пролиферације ћелија са брзином 28 микрометара по сату а) иницијална колонија б) после 5 дана в) после 8 дана г) после 10 дана

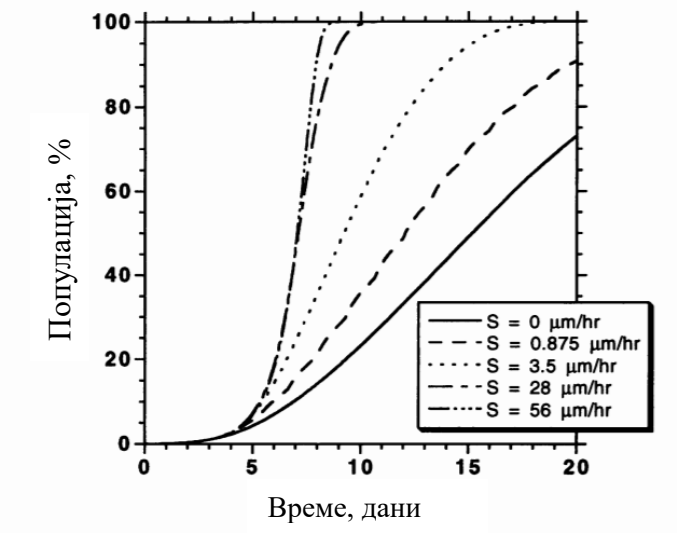


Слика 19 Симулација пролиферације ћелија са брзином 56 микрометара по сату а) иницијална колонија б) после 2 дана в) после 6 дана г) после 8 дана



Слика 20 Функција раста колоније ћелија у зависности од времена, брзина 56 микрометара по сату

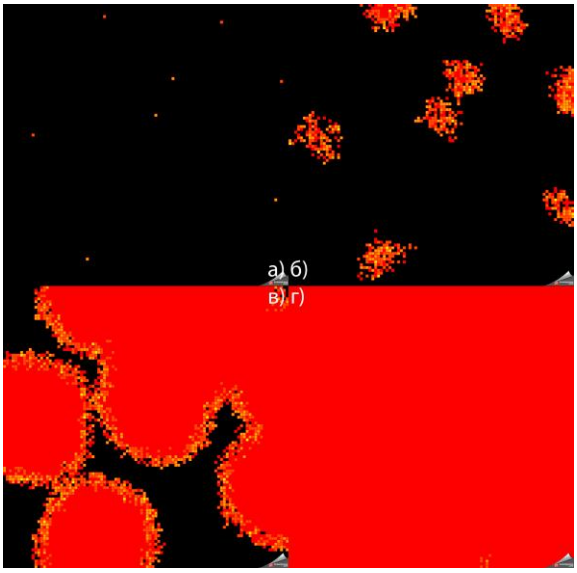
У раду [8], сличан експеримент је извршен чији су резултати приказани на слици 21.



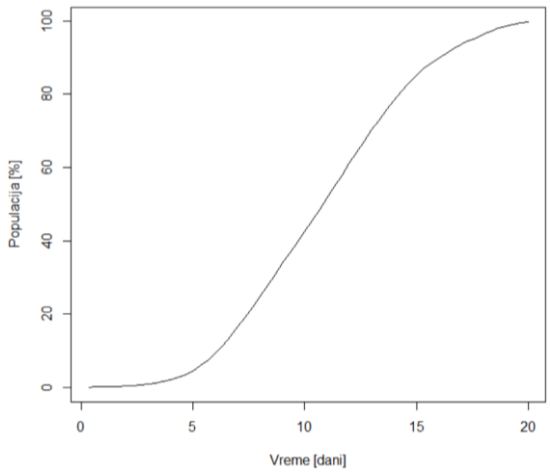
Слика 21 Фунцкија раста колонија ћелија у зависности од времена и брзина [1]

Резултати добијени ћелијским аутоматом описаним у овом раду и резултати из [1] се поклапају за брзине од 28 и 56 микрометара по сату. За ниже брзине, долази до одступања приближно 20 %. Ова драстична разлика се испољава због конфигурације иницијалне колоније, где се у [1] користи насумично распоређивање иницијалне популације, док се у симулацији описаној у овом раду користи концентрисана популација у центру. Променом иницијалног распореда из популације концентрисане у центру, на насумичне позиције унутар матрице за исту почетну популацију, време потребно за стварање конфлуентног слоја се смањује за готово 20%, са 25 дана на 20. Разлог томе се огледа у могућности кретања ћелија у почетним тренуцима када су ћелије

насумично распоређене у односу на густо паковање ћелија у централни део матрице, где унутрашње ћелије немају могућност кретања и деобе. Резултати симулације насумично распоређене иницијалне популације, за брзину од 3.5 микрометара по сату су приказани на слици 22, а раст популације кроз време је приказан на слици 23.



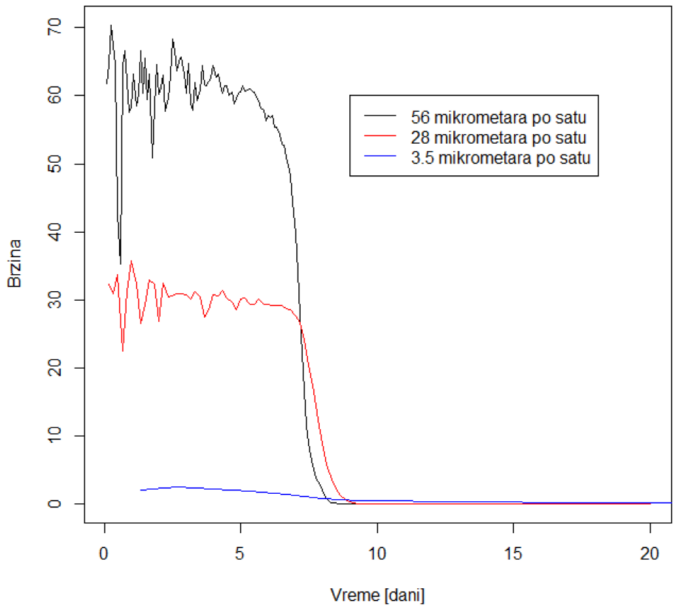
Слика 22 Симулација пролиферације ћелија са просечном брзином 3.5 микрометара по сату и насумичном поставком иницијалне популације а) Иницијална колонија б) Колонија после 5 дана в) Колонија после 15 дана г) Колонија после 20 дана



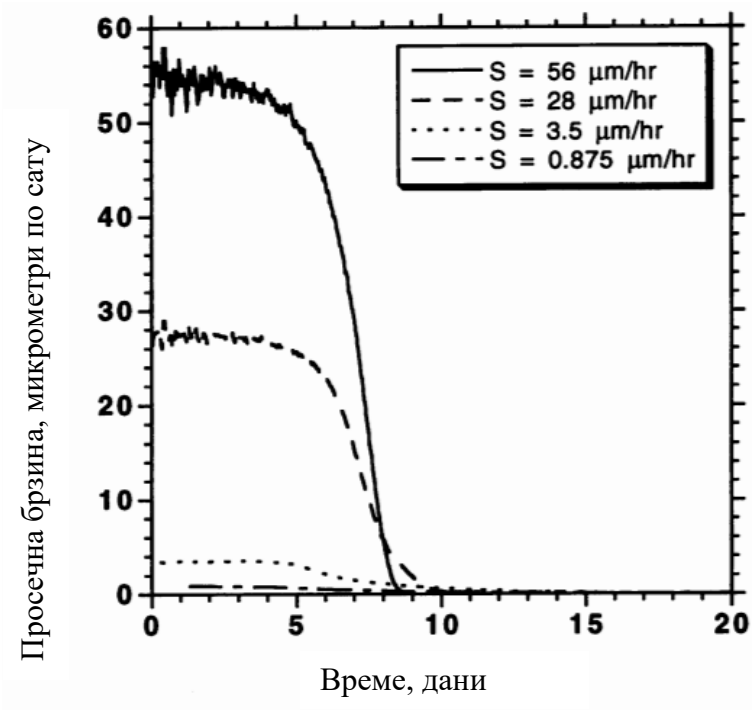
Слика 23 Функција раста колоније ћелија у зависности од времена, брзина 56 микрометара по сату, насумична поставка иницијалне популације

Како популација расте, појединачне ћелије имају мање простора за миграцију због контактне ихибиције, те се просечна брзина популације смањује растом. Овај феномен је приказан на слици 24, где се може видети пад просечне брзине колоније у односу на време. На слици 25 је приказан резултат истог експеримента добијен у раду [1]. Поређењем слике 24 и 25, може се закључити да долази до скоро потпуног поклапања

резултата. Разлике које се могу уочити, производ су стохастике присутне у ћелијском аутомату, како у иницијалној популацији, тако и у времену деобе и смеру кретања.



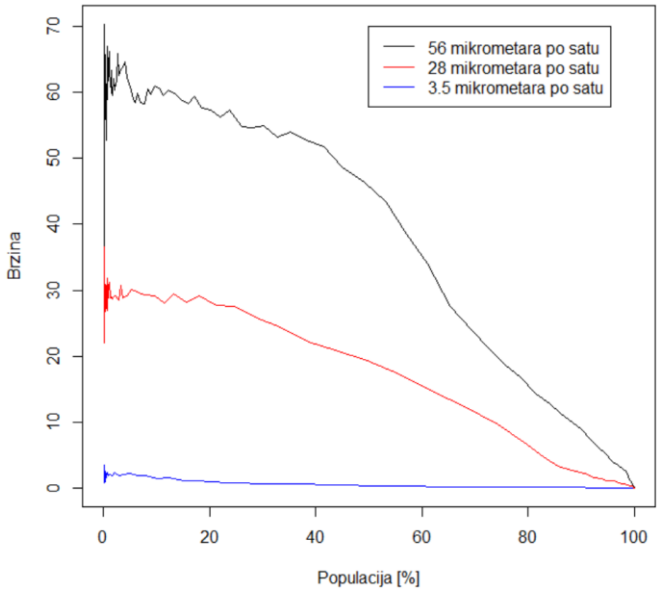
Слика 24 Зависност просечне брзине популације у односу на време, брзине ћелија приказане су у легенди



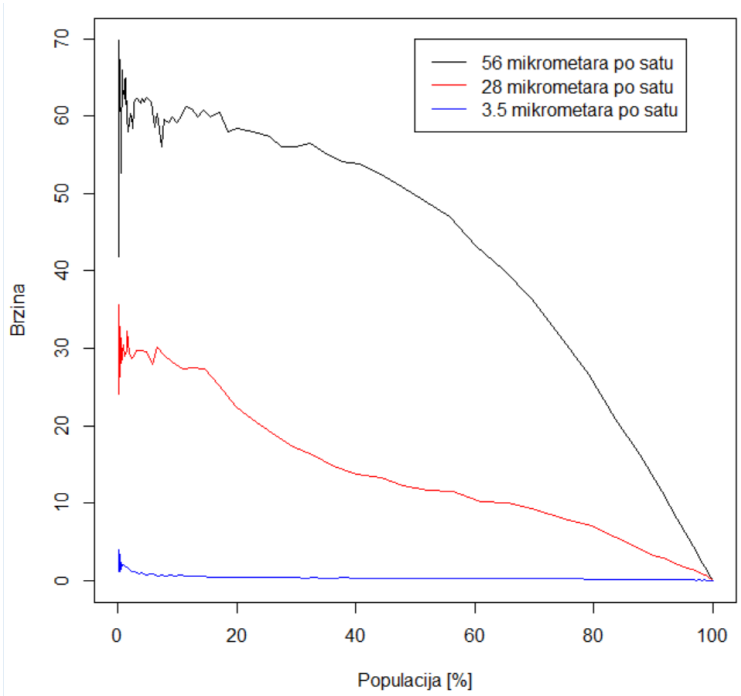
Слика 25 Просечне брзине популације, добијене у раду [1]

Односи просечних брзина и величина популација за насумичан распоред иницијалне популације су приказани на слици 26. На слици 27, приказан је однос просечних брзина и величина популација за иницијалну популацију концентрисану у

средишту. Са слике 26 и 27, закључује се да иницијални распоред нема велики утицај на функцију брзине од величине популације.

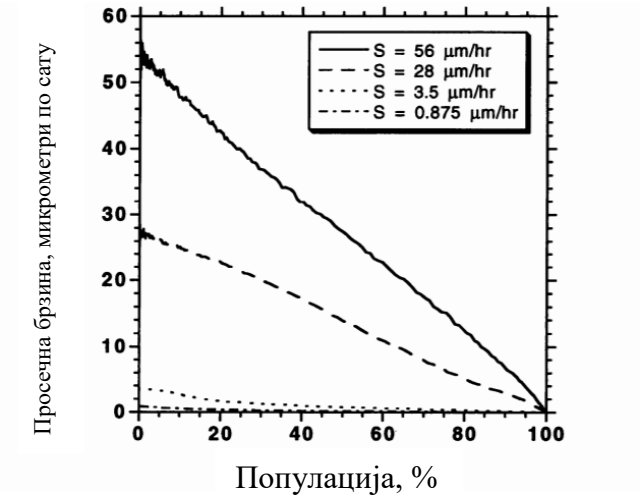


Слика 26 Просечна брзина у односу на величину популације, насумични иницијални распоред



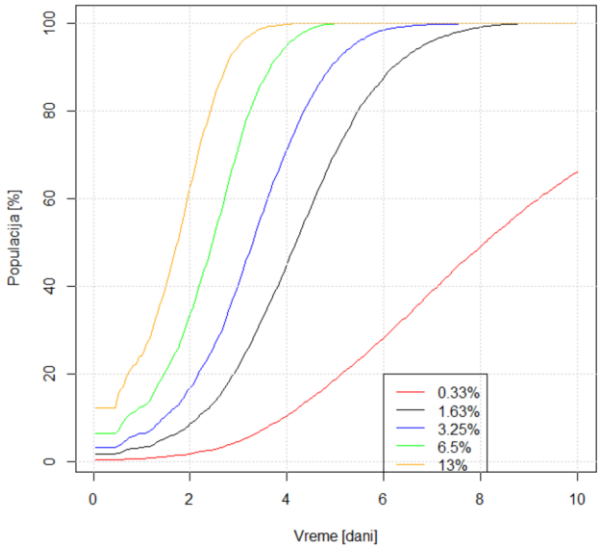
Слика 27 Просечна брзина у односу на величину популације, иницијални распоред концентрисан у средини

Слични резултати су добијени и у раду [1], и приказани су на слици 28.

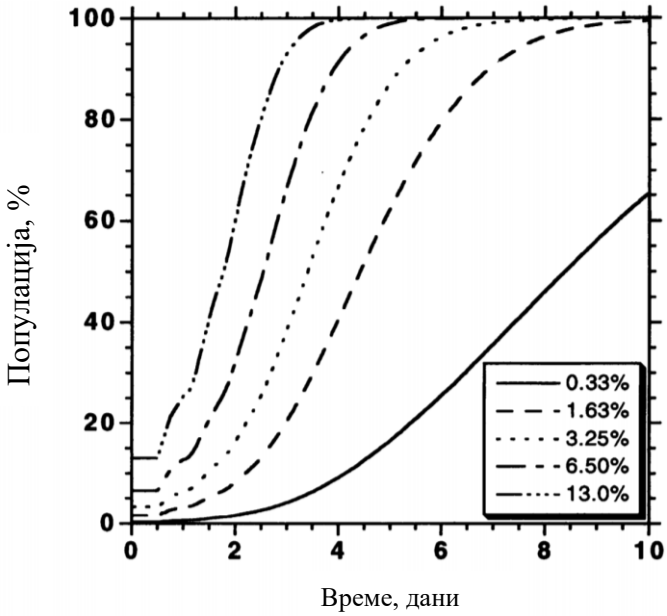


Слика 28 Просечна брзина у односу на величину популације, добијена у раду [1]

У следећој симулацији, варирана је иницијална величина популације, која је концентрисана у средини матрице, и проверавано је како она утиче на брзину испуњавања целе матрице. Резултат је представљен на слици 29. На слици 30, представљен је добијани резултат у раду [1]. Са слика 29 и 30, може се видети да се резултати потпуно поклапају.



Слика 29 Зависност брзине популације у односу на величину иницијалне популације



Слика 30 Зависност брзине популације у односу на величину иницијалне популације, добијена из рада [1]

Закључак

У овом раду, описано је понашање пролиферација миграторних ћелија које испољавају контактну инхибицију. Објашњена је њихова важност у формирању ожиљачног ткива, ендотела артерија, механизам стварања тумора. Показан је моћни апарат за симулацију природних појава у облику ћелијског аутомата. Пролиферација миграторних ћелија је потом симулирана коришћењем овог апарата, притом објашњавајући детаље имплементације симулације, зарад лаке репродукције резултата добијених овим радом. Симулације су вршене за различите конфигурације иницијалне популације и брзина, где су мерени релевантни параметри попут просечне брзине популације, раст популације ћелија кроз време, као и међусобне зависности ове две величине. Добијени резултати су верификовани резултатима добијених претходним радовима у овом пољу, који показују високи степен поклапања.

Референце

[1] Yih Lee, Stylianos Kouvroukoglou, Larry V. McIntire, and Kyriacos Zygourakis, *A Cellular Automaton Model for the Proliferation of Migrating Contact-inhibited Cells*, Biophysical Journal Volume 69, October 1995

[2] Lee, Y., *Computer-assisted analysis of endothelial cell migration and proliferation. Ph.D. thesis*, Rice University, Houston, TX, 1994

[3] Mak, M.; Spill, F.; Roger, K.; Zaman, M., *Single-Cell Migration in Complex Microenvironments: Mechanics and Signaling Dynamics*, Journal of Biomechanical Engineering. 138 (2): 021004. doi:10.1115/1.4032188. PMC 4844084. PMID 26639083.).

[4] Creative Bioarray: *Endothelial cells and media*, https://www.creative-bioarray.com/filter/endothelial-cell-and-media-10.html?gclid=CjwKCAjw-5v7BRAmEiwAJ3DpuLPoyTSC9FFvaB-P-QOFaxjUQF1maFBkMwgaH2oFYS27BGjRlzGT_RoCL70QAvD_BwE#flilteртop, 2020.

[5] Andras Czirok, *Endothelial cell motility, coordination and pattern formation during vasculogenesis*, Wiley Interdiscip Rev Syst Biol Med., 2013 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3737767/#:~:text=2.1%20Motility%20of%20individual%20cells,speed%20along%20a%20straight%20line

[6] Takeshi Kawauchi, *Cell Adhesion and Its Endocytic Regulation in Cell Migration during Neural Development and Cancer Metastasis*, International Journal of Molecular Sciences, 2012 – ISSN 1422-0067

[7] Stokes CL, Lauffenburger DA, Williams SK., *Migration of individual microvessel endothelial cells: stochastic model and parameter measurement*., J Cell Sci. 1991;99:419–30. [PubMed]

[8] Grotendorst, G. R., H. Seppa, H. K. Kleinman, and G. R. Martin., *Attachment of smooth muscle cells to collagen and their migration toward platelet-derived growth factor*, Proc. Natl. Acad. Sci. USA, 1981, 78:3669-3672.

[9] Sato, Y., D. B. Rifkin., *Autocrine activities of basic fibroblast growth factor: regulation of endothelial cell movement, plasminogen activator synthesis, and DNA synthesis*., J. Cell Biol., 1988. 107: 1199-1205.

[10] Chiara De Pascalisa,b, Sandrine Etienne-Manneville, *Single and collective cell migration: the mechanics of adhesions*, Molecular Biology of the Cell., 2017, doi: 10.1091/mbc.E17-03-0134

[11] Y. Lee, L.V. Mclntire, and K. Zygourakis, *Analysis of Endothelial Cell Locomotion: Differential Effects of Motility and Contact Inhibition*, Department of Chemical Engineering and Institute of Biosciences and Bioengineering, Rice University, Houston, Texas, 1993, https://onlinelibrary.wiley.com/doi/epdf/10.1002/bit.260430712

[12] Berto, Francesco and Jacopo Tagliabue, "Cellular Automata", *The Stanford Encyclopedia of Philosophy* (Fall 2017 Edition), Edward N. Zalta (ed.), 2017, https://plato.stanford.edu/entries/cellular-automata/

[13]Dmitry Zaitsev, *k-neighborhood for Cellular Automata*, Faculty of Engineering Vistula University Stoklosy 3, 02-787 Warszawa, Poland, 2016,

[14]Stephen Wolfram, *Cellular automata*, https://content.wolfram.com/uploads/sites/34/2020/07/cellular-automata.pdf

[15]Daniel Shiffman, Nature of code, https://natureofcode.com/book/chapter-7-cellular-automata/

[16]Stephen Wolfram, *Cellular automata as models of complexity*, The Institute for Advanced Study, Princeton, New Jersey 08510, USA, 1984 https://content.wolfram.com/uploads/sites/34/2020/07/cellular-automata-models-complexity.pdf

[17]Yih Lee, Larry V. Mclntire, Kyriacos Zygourakis, Pauline A. Markenscoff, *Characterization of endothelial cell locomotion using a Markov chain model*, Biochemistry and Cell Biology, 1995

[18]K. Zygourakis R. Bizios P. Markenscoff, *Proliferation of anchorage-dependent contact-inhibited cells: I. Development of theoretical models based on cellular automata,* Bioingeneering and biotechnology, 1991 https://doi.org/10.1002/bit.260380504

[19]Stack Overflow, *Generate random number between 0 and 1 with (negative)exponential distribution*, https://stackoverflow.com/a/20408401, Септембар 2020

Прилог 1 – Једнодимензиони ћелијски аутомат

```
 1 int[] cells = new int[64];
 2 int timeStep = 0;
 3 void setup(){
 4   size(640,320);
 5   frameRate(10);
 6   background(0);
 7   for (int i = 0; i < 64; i++){
 8     if (i != 31){
 9       cells[i] = 0;
10     } else {
11       cells[i] = 1;
12     }
13   }
14 }
15
16 void draw(){
17   println(timeStep);
18   drawCells();
19   compute();
20   timeStep++;
21 }
22
23 void compute(){
24   int[] nextState = new int[64];
25   arrayCopy(cells, nextState);
26   for (int i = 1; i < 63; i++){
27     nextState[i] = (cells[i-1] + cells[i+1]) % 2;
28   }
29   arrayCopy(nextState, cells);
30 }
31 void drawCells(){
32   for (int i = 0; i < 64; i++){
33     fill(255);
34     if (cells[i] > 0){
35       rect(i*10, timeStep * 10, 10, 10);
36     }
37   }
38 }
```

Прилог 2 – Дводимензиони ћелијски аутомат

```
int[][] currentState = new int[30][30];
int k = 0;
void setup(){
  size(600,600);
  frameRate(10);
  initialize();
}

void initialize(){
  for (int i = 0; i < 30; i++){
    for (int j = 0; j < 30; j++){
      currentState[i][j] = int(random(2));
    }
  }

}

void draw(){
  drawCells();
  compute();
}


void compute(){
  int[][] nextState = copyMatrix(currentState);
  for (int i = 0; i < 30; i++){
    for (int j = 0; j < 30; j++){
      int count = countNeighbours(i,j);
      switch (currentState[i][j]){
        case 0:
          if (count == 3){
            nextState[i][j] = 1;
          } else {
            nextState[i][j] = 0;
          }
          break;
        case 1:
          if (count > 3 || count < 2){
            nextState[i][j] = 0;
          } else {
            nextState[i][j] = 1;
          }
          break;
      }
    }
  }
  currentState = copyMatrix(nextState);
}

int countNeighbours(int x, int y){
  int count = 0;
  for (int i = x - 1; i <= x + 1; i ++){
    for (int j = y - 1; j <= y + 1; j ++){
      if (i >= 0 && i < 30 && j >= 0 && j < 30 && !(i == x && j == y)){
```

```
55          count += currentState[i][j];
56        }
57      }
58    }
59    return count;
60  }
61
62  int[][] copyMatrix(int[][] matrix){
63    int[][] newMatrix = new int[30][30];
64    for (int i = 0; i < 30; i++){
65      for (int j = 0; j < 30; j++){
66         newMatrix[i][j] = matrix[i][j];
67      }
68    }
69    return newMatrix;
70  }
71
72  void drawCells(){
73    fill(255);
74    background(0);
75    for (int i = 0; i < 30; i++){
76      for (int j = 0; j < 30; j++){
77         if (currentState[i][j] == 1){
78           rect(i*20, j*20, 20, 20);
79         }
80      }
81    }
82  }
83
84
```

Прилог 3 – Ћелијски аутомат за симулацију пролиферације миграторних ћелија са контаткном инхибицијом

```
import java.util.*;
//matrica trenutnog stanja, u njoj se nalaze sve ćelije koje se
trenutno vide
int[][] playground;
//indeksi kolona i redova po kojim će se redom pristupati odabiru
trenutne ćelije
//umesto da radimo n*n random pozivanja za svaku ćeliju, možemo da
optimizujemo na 2n, tako što ćemo randomizovati niz redova i kolona
//te onda pristupati redom članovima
int[][] indices;
//da li je simulacija stopirana ili pokrenuta
boolean run = false;
//trenutno vreme simulacije
double t;
//trenutni korak simulacije
int k = 0;
//random generator
Random random;
//trenutna ukupna ćelijska populacija
int population = 0;
int[] divisionDistribution = new int[100];
double distance = 0;
List<Integer> directionDistribution = new ArrayList();
double rate = 0.5;
int size = 10000;
void populateDivisionDistributionArray(){
  for (int i = 0; i < 100; i++){
    if (i < 64){
      divisionDistribution[i] = (int)((12f +int(random(6))) / rate);
    } else if (i < 96){
      divisionDistribution[i] = (int)((18f + int(random(6))) / rate);
    } else {
      divisionDistribution[i] = (int) ((24f + int(random(6))) / rate);
    }
    divisionDistribution[i] = min(99, divisionDistribution[i]);
  }
}
void setup() {
  frameRate(250);
  t = 0;
  size(1200, 1200);
  colorMode(HSB, 19, 100, 10);
  //  frameRate(10);
  random = new Random();
  populateDivisionDistributionArray();
  playground = new int[100][100];
  for (int i = 0; i < 100; i++) {
    for (int j = 0; j < 100; j++) {
      playground[i][j] = -1;
    }
  }

```

```
53   //postavljenje 32 celije u centar matrice
54   for (int i = 0; i < 100; i ++) {
55     for (int j = 0; j < 100; j ++) {
56       if ((i - 49) * (i - 49) + (j - 49) * (j - 49) < 10) {
57         playground[i][j] = createCell();
58     }
59   }
60   //postavljanje 1200 celija na nasumicna mesta
61   //for (int i = 0; i < 1200; i++){
62   //  playground[(int)random(100)][(int)random(100)] = createCell();
63   //}
64   //playground[50][50] = createCell();
65   indices = new int[2][100];
66   for (int i = 0; i < 2; i++) {
67     for (int j = 0; j < 100; j ++) {
68       indices[i][j]=j;
69     }
70     shuffle(indices[i]);
71   }
72 }
73 float logRandom(){
74   return -log(1 - (1 - exp(-6)) * random(1)) / 6;
75 }
76 int createCell(){
77   //odabir pravca
78   int direction = int(random(9)) * 1000;
79   //odabir vremena kretanja u tom pravcu, poštujući inverznu
80 eksponencijalnu distribuciju
81   int persistenceCounter = min(9, int((int)(logRandom() * 5. / rate)))
82 * 100;
83   //odabir vremena do deobe, poštujući distribuciju vremena
84   int divisionCounter = divisionDistribution[int(random(100))];
85   return direction+persistenceCounter+divisionCounter;
86 }
87 void draw() {
88   if (run) {
89     compute();
90     if (k++ % 2 == 0){
91     //println(t/24.0 + "," + (t == 0 ? 0 :
92 (distance/(double)population)/rate));
93     //println(d + "," + population);
94     println(t / 24.0 + "," + (double)(population/(double)size) * 100);
95    // println(((double)(population/10000.) * 100) + "," + (t == 0 ? 0 :
96 ((distance/(double)population)/rate)));
97   }
98     t+= rate;
99   }
100  drawCells();
101 }
102 void keyReleased() {
103   run = !run;
104 }
105 void compute() {
106   int[][] nextState = new int[100][100];
107   //arrayCopy(playground, nextState);
108   for (int i = 0; i < 100; i++) {
109     for (int j = 0; j < 100; j++) {
```

```
110       arrayCopy(playground[i], nextState[i]);
111     }
112   }
113   population = 0;
114   distance = 0;
115   d = 0;
116   for (int i = 0; i < 100; i++) {
117     for (int j = 0; j < 100; j++) {
118       int currentCell = playground[indices[0][i]][indices[1][j]];
119       if (currentCell > -1 && currentCell < 10000) {
120         d++;
121         if (divisionCount(currentCell) == 0) {
122           if (divide(indices[0][i], indices[1][j], nextState)) {
123             continue;
124           }
125         }
126         if (directionCount(currentCell) == 0) {
127           int dir = direction(currentCell);
128           List<Integer> freeDirections = new ArrayList();
129           for (int t = 1; t < 9; t++) {
130             if (freeDirection(t, indices[0][i], indices[1][j],
131 nextState)) {
132               freeDirections.add(t);
133             }
134           }
135           dir = newDirection(dir, freeDirections);
136           int div = divisionCount(currentCell);
137           //div -= div > 0 ? 1 : 0;
138           if (dir > 0) {
139             currentCell = dir * 1000 +  min(9, int((int)(logRandom() *
140 5. / rate))) * 100 + div;
141           } else {
142             currentCell = dir * 1000 + 0 * 100 + div;
143           }
144         }
145         migrate(currentCell, indices[0][i], indices[1][j], nextState);
146       }
147     }
148   }
149   for (int i =0; i < 100; i++){
150     for (int j = 0; j < 100; j++){
151       if (nextState[i][j] > - 1 && nextState[i][j] < 10000){
152         population++;
153       }
154     }
155   }
156   arrayCopy(nextState, playground);
157   shuffle(indices[0]);
158   shuffle(indices[1]);
159 }
160 int newDirection(int oldDirection, List<Integer> possibleDirections){
161    directionDistribution.clear();
162   //ako ćelija stoji u mestu, 80 posto je šansa da ostane u mestu
163   //ako ne stoji u mestu, šansa da nastavi kretanje u prethodnom smeru
164 je 5%
165   if (oldDirection == 0){
166     if ((int)random(10)>1){
```

```
167       return 0;
168     } else {
169       if (possibleDirections.isEmpty()){
170         return 0;
171       } else {
172         return possibleDirections.get((int)
173 random(possibleDirections.size()));
174       }
175     }
176   } else if (possibleDirections.contains(oldDirection)){
177     directionDistribution.add(oldDirection);
178   } else if (possibleDirections.isEmpty()){
179     return 0;
180   }
181   //+- 45 stepeni
182   addDirections(getDirections(oldDirection,1, possibleDirections), 97);
183   //+- 90 stepeni
184   addDirections(getDirections(oldDirection,2,possibleDirections), 20);
185   //+-135 stepeni
186   addDirections(getDirections(oldDirection,3,possibleDirections), 8);
187   //+-180 stepeni
188   addDirections(getDirections(oldDirection,4,possibleDirections), 10);
189   return
190 directionDistribution.get((int)random(directionDistribution.size()));
191 }
192 List<Integer> directions = new ArrayList();
193 List<Integer> getDirections(int oldDirection, int newDirection,
194 List<Integer> possibleDirections){
195   directions.clear();
196   int direction = abs(8 + oldDirection - newDirection) % 8;
197   if (direction == 0){
198     direction = 8;
199   }
200   if (possibleDirections.contains(direction)){
201     directions.add(direction);
202   }
203   direction = abs(8 + oldDirection + newDirection) % 8;
204   if (direction == 0){
205     direction = 8;
206   }
207   if (possibleDirections.contains(direction)){
208     directions.add(direction);
209   }
210   return directions;
211 }
212 void addDirections(List<Integer> directions, int count){
213   if (directions.isEmpty()){
214     return;
215   }
216   for (int i = 0; i < count; i++){
217     if (i < count/directions.size()){
218       directionDistribution.add(directions.get(0));
219     } else {
220       directionDistribution.add(directions.get(1));
221     }
222   }
223 }
```

```
boolean freeDirection(int dir, int i, int j, int[][] nextState) {
  switch(dir) {
  case 0:
    return true;
  case 1:
    if (i+1 < 100 && nextState[i+1][j] == -1 ) {
      return true;
    }
    break;
  case 2:
    if (i+1 < 100 && j-1 >= 0 && nextState[i+1][j-1] == -1) {
      return true;
    }
    break;
  case 3:
    if (j-1 >= 0  && nextState[i][j-1] == -1) {
      return true;
    }
    break;
  case 4:
    if (j-1 >= 0 && i - 1 >= 0 && nextState[i-1][j-1] == -1) {
      return true;
    }
    break;
  case 5:
    if (i - 1 >= 0 && nextState[i-1][j] == -1) {
      return true;
    }
    break;
  case 6:
    if (j + 1 < 100 && i - 1 >= 0 && nextState[i-1][j+1] == -1) {
      return true;
    }
    break;
  case 7:
    if (j + 1 < 100 && nextState[i][j+1] == -1) {
      return true;
    }
    break;
  case 8:
    if (j + 1 < 100 && i + 1 < 100 && nextState[i+1][j+1] == -1) {
      return true;
    }
    break;
  }
  return false;
}
int d = 0;
void migrate(int currentCell, int i, int j, int[][] nextState) {
  int div = divisionCount(currentCell);
  int dirCnt = directionCount(currentCell);
  nextState[i][j] = -1;
  //if (direction(currentCell) != 0 && currentCell > -1){
  //  d++;
  //}
  switch(direction(currentCell)) {
  case 0:
```

```
281     break;
282   case 1:
283     if (i+1 < 100 && nextState[i+1][j] == -1) {
284       i+=1;
285       distance+=28;
286     } else {
287       dirCnt = 0;
288     }
289     break;
290   case 2:
291     if (i+1 < 100 && j-1 >= 0 && nextState[i+1][j-1] == -1) {
292       i += 1;
293       j -= 1;
294       distance+=1.41*28;
295     } else {
296       dirCnt = 0;
297     }
298     break;
299   case 3:
300     if (j-1 >= 0  && nextState[i][j-1] == -1) {
301       j-=1;
302       distance+=28;
303     } else {
304       dirCnt = 0;
305     }
306     break;
307   case 4:
308     if (j-1 >= 0 && i - 1 >= 0 && nextState[i-1][j-1] == -1) {
309       i -= 1;
310       j -= 1;
311       distance+=1.41*28;
312     } else {
313       dirCnt = 0;
314     }
315     break;
316   case 5:
317     if (i - 1 >= 0 && nextState[i-1][j] == -1) {
318       i -= 1;
319       distance+=28;
320     } else {
321       dirCnt = 0;
322     }
323     break;
324   case 6:
325     if (j + 1 < 100 && i - 1 >= 0 && nextState[i-1][j+1] == -1) {
326       j += 1;
327       i -= 1;
328       distance+=1.41*28;
329     } else {
330       dirCnt = 0;
331     }
332     break;
333   case 7:
334     if (j + 1 < 100 && nextState[i][j+1] == -1) {
335       j += 1;
336       distance+=28;
337     } else {
```

```
338       dirCnt = 0;
339     }
340     break;
341   case 8:
342     if (j + 1 < 100 && i + 1 < 100 && nextState[i+1][j+1] == -1) {
343       j += 1;
344       i += 1;
345       distance+=1.41*28;
346     } else {
347       dirCnt = 0;
348     }
349     break;
350   }
351   div -= div > 0 ? 1 : 0;
352   dirCnt -= dirCnt > 0 ? 1 : 0;
353   nextState[i][j] = direction(currentCell) * 1000 + dirCnt * 100 + div;
354 }
355 List<Integer> freeCellX = new ArrayList();
356 List<Integer> freeCellY = new ArrayList();
357 boolean divide(int i, int j, int[][] nextState) {
358   freeCellX.clear();
359   freeCellY.clear();
360   int newX = 0;
361   int newY = 0;
362   boolean found = false;
363   for (int x = -1; x < 2; x ++) {
364     for (int y = -1; y < 2; y ++) {
365       if (i + x >= 0 && j + y >= 0  && i + x < 100 && j + y < 100 &&
366 nextState[i+x][j+y] == -1) {
367         //if ((i+j)%2 == 0) continue;
368         if (i+x == i && j+y == j) continue;
369         freeCellX.add(i+x);
370         freeCellY.add(j+y);
371         if (found) {
372           if (int(random(2)) == 0) {
373             newX = i+x;
374             newY = j+y;
375           }
376         } else {
377           found = true;
378           newX = i+x;
379           newY = j+y;
380         }
381       }
382     }
383   }
384   if (!freeCellX.isEmpty()) {
385     int index = random.nextInt(freeCellX.size());
386     //nextState[newX][newY] = createCell();
387     nextState[freeCellX.get(index)][freeCellY.get(index)] =
388 createCell();
389   }
390   nextState[i][j] = createCell();
391   return found;
392 }
393
394 void drawCells() {
```

```
395   background(0);
396   for (int i = 0; i < 100; i++) {
397     for (int j = 0; j < 100; j++) {
398       if (playground[i][j] != -1) {
399         if (playground[i][j] < 10000) {
400           stroke(255-divisionCount(playground[i][j]), 100, 10 -
401 directionCount(playground[i][j]));
402           fill(divisionCount(playground[i][j]), 100, 10 -
403 directionCount(playground[i][j]));
404           rect(i*12, j*12, 12, 12);
405         } else {
406           stroke(100, 0, 100);
407           fill(100, 0, 100);
408         }
409         rect(i*12, j*12, 12, 12);
410       }
411     }
412   }
413 }
414 int direction(int state) {
415   return state / 1000;
416 }
417 int directionCount(int state) {
      return ((state % 1000) / 100);
    }
    int divisionCount(int state) {
      return state % 100;
    }

    void shuffle(int[] array) {
      for (int i = 0; i < array.length; i++) {
        int index = random.nextInt(i + 1);
        if (index != i)
        {
          array[index] ^= array[i];
          array[i] ^= array[index];
          array[index] ^= array[i];
        }
      }
    }
```