hochschule mannheim

Extension of two dimensions morphogenesis simulation models of the urothelium into three dimensions within the moduro simulation environment

Thorsten Mueller

Bachelor Thesis

for the acquisition of the academic degree Bachelor of Science (B.Sc.)

Course of Studies: Computer Science

Department of Computer Science

University of Applied Sciences Mannheim

11.11.2015

Tutors

Prof. Dr. Markus Gumbel, Hochschule Mannheim

Erika Mustermann, Paukenschlag GmbH

Mueller, Thorsten:
Extension of two dimensions morphogenesis simulation models of the urothelium into three dimensions within the moduro simulation environment / Thorsten Mueller. –
Bachelor Thesis, Mannheim: University of Applied Sciences Mannheim, 2018. 38 pages.

Mueller, Thorsten:
Einsatz eines Flux-Kompensators für Zeitreisen mit einer maximalen Höchstgeschwindigkeit von WARP 7 / Thorsten Mueller. –
Bachelor-Thesis, Mannheim: Hochschule Mannheim, 2018. 38 Seiten.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Mannheim öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Mannheim, 11.11.2015

Thorsten Mueller

Abstract

Extension of two dimensions morphogenesis simulation models of the urothelium into three dimensions within the moduro simulation environment

The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary. The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to pay expensive translators. To achieve this, it would be necessary to have uniform grammar, pronunciation and more common words. If several languages coalesce, the grammar of the resulting language is more simple and regular than that of the individual languages. The new common language will be more simple and regular than the existing European languages. It will be as simple as Occidental; in fact, it will be Occidental. To an English person, it will seem like simplified English, as a skeptical Cambridge friend of mine told me what Occidental is.

Einsatz eines Flux-Kompensators für Zeitreisen mit einer maximalen Höchstgeschwindigkeit von WARP 7

Jemand musste Josef K. verleumdet haben, denn ohne dass er etwas Böses getan hätte, wurde er eines Morgens verhaftet. Wie ein Hund! sagte er, es war, als sollte die Scham ihn überleben. Als Gregor Samsa eines Morgens aus unruhigen Träumen erwachte, fand er sich in seinem Bett zu einem ungeheueren Ungeziefer verwandelt. Und es war ihnen wie eine Bestätigung ihrer neuen Träume und guten Absichten, als am Ziele ihrer Fahrt die Tochter als erste sich erhob und ihren jungen Körper dehnte. Es ist ein eigentümlicher Apparat, sagte der Offizier zu dem Forschungsreisenden und überblickte mit einem gewissermaßen bewundernden Blick den ihm doch wohl bekannten Apparat. Sie hätten noch ins Boot springen können, aber der Reisende hob ein schweres, geknotetes Tau vom Boden, drohte ihnen damit und hielt sie dadurch von dem Sprunge ab. In den letzten Jahrzehnten ist das Interesse an Künstlern sehr zurückgegangen. Aber sie überwanden sich, umdrängten den Käfig und wollten sich gar nicht fortrühren.

Contents

Chapter 1

Introduction

1.1. Motivation

Without cell adhesion no complex life would exist **REF** This is a well known and long existing statement in the biology. It describes that if different cells would not stick to each other the only living things would be cells. In humans, organs are made of severall cells. This is also the case for the urothelium. For the urothelium it is necessary that the cells stick to each other. Otherwise the functions of the urothelium could not be executed and also it would not be able to growth.

There are two types of tumors. One is the benign and the other one is the malignant tumor [1]. The benign tumor is self limited. Thus, it does not invade surrounding tissues as well as it does not spread into other body parts [1]. The malignant tumor on the other hand is not limited in its growth and is able to invade other body parts [1]. Since bladder cancer is one of the most common cancer types among men it is important to understand how and why the cancer is able to growth. Bladder cancer starts to growth in the urothelium. With the growth and spread the structure of cells sticking together is changed. In this case, the urothelium is not able to completely do its tasks. In order to understand the urothelium and how and when bladder cancer appears observations of this organ are necessary.

To understand the functionality of organs observation is necessary. For the urothelium this is already done, as there are a lot of different in vitro experiments about the methodology of the urothlium. After the observation of organs is complete, we are able to predict how the organ will react to different situations. To verify these predictions we need to simulate the behavior of the organ. A simulation is in generell an illustration of the reality, but it can also be used to change reality in a for the research specific way to get more knowledge of this organ.

A simulation should always be a simple as possible but also not to simple. Otherwise the simulation does not displays the reality. In the moduro project there is a 2D simulation of 16 different models of the urothelium. After all them were simulated the result is that the models are more realistic and others are less. The target with these models is to predict when bladder cancer occurs.

Because a cell is a three dimensional organism, a 2D simulation of the urothelium might not give as many aspects as a 3D simultion could do. The aim of this bachelor thesis is to create a 3D simulation of these 16 different models. With this 3D simulation new insights of the urothelium and how bladder cancer occurs will be received.

1.2. Background

1.2.1. Biology of the Urothelium

Bladder cancer is the 4th most common cancer type in men regarding to everydayhealth.com [2], where every 36st out of 100.000 men gets it. Bladder cancer usually starts with some cells in the bladder, which growth uncontrolled. From these cells, the tumor can spread further into surrounding areas [3]. The most common bladder cancer type is the urothelial carcinoma [3]. An illustration of how bladder cancer evolves is displayed in figure 1.1 at page 3.

The bladder is located in the lower urinary tract and consist of several parts, where urothelium coats the bladder [4]. More specifically, it covers the bladder from the renal pelvis to the proximal urethra [5], [6]

Two important tasks of the bladder are the storage and release of urine. To do so the bladder will extend, during the storage, and then shrink again [7]. One task of the urothelium is to form a distensible barrier [4], [8]–[11], which prevents unregulated exchange of ions, solutes, and toxic metabolites between the bladder and the blood [4], [8]–[10]. That the urothelium ensures its barrier function, it has to growth and downsize its size. This is done by the largest cells of the urothelium – the umbrella cells. Because the umbrella cells are in direct contact with the bladder, it is their task to change size and form during the growth and shrink process of the

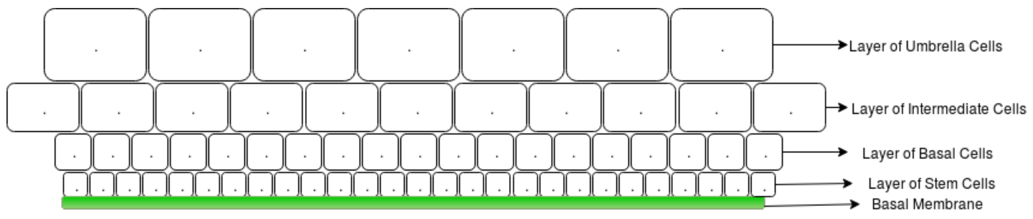

Figure 1.1.: Simplified physiology of the urothelium

bladder. Birder described the urothelium as "... a responsive structure capable of detecting physiological and chemical stimuli and releasing a number of signaling mole-cules." [6]. Another task of the urothelium is to control the movement and passage of macromolecules, ions, water, toxic metabolites and solutes [8], [9]. If the urothelium is damaged, it rapidly generates new cells, to ensure full functionality [5], [8], [9]. In the research there are assumptions, that the quick regeneration is done by a lot of mitosis, i.e. cell division.

To receive a better overview the different cell types are explained in the following paragraph. In figure 1.1 at page 3 a simplified illustration of the urothelium with its different cell types and cell layers is provided.
The umbrella cells, also called superficial cells, they are connected directly with the bladder and have an average diameter of 25 up to $250\,\mu\mathrm{m}$ [5], [9].

Below these cells the intermediate cells are located. With an average diameter of 10 up to $20\,\mu\mathrm{m}$ [5], [9], they are a far bit smaller then the umbrella cells. The middle layer of the urothelium can consist of several own layers, i.e. there can be several layers in the layer of the intermediate cells.

The smallest and the most common cells in the urothelium are the basal and stem cells. Those cells have a diameter of up to $10\,\mu\mathrm{m}$ [4], [9].

The urothelium has several layers. In the first layer, there are the basal and stem cells. Above them, there are several layers of intermediate cells. On top of the epithelium, i.e. a membranous tissue which consists of one or several layers1, there is one layer of umbrella cells.

1.2.2. Glazier Graner Hogeweg Model

Since there are several formulas and models developed from Glazier and Graner, this subsection briefly describes these models and formulars.

Glazier-Graner-Hogeweg (GGH) models are widely used in biological simulations, since it provides a good flexibility, extensibility and it is easy to use [12]. Glazier and Graner developed their model as an extension of the large-q Potts model, which itself is an extension of the Ising Model, and called it first the Extended Potts Model (EPM). Nowadays, this model is called Cellular Potts Model (CPM) [12]–[14]. Glazier and Graner extended their CPM in a way that also volume constraints are considered for the hamiltonian, see following form:

$$\begin{aligned}\mathcal{H}_{CPM} = &\sum_{\vec{i},\vec{j}} J(\tau(\sigma(\vec{i})),(\tau(\sigma(\vec{j})))(1-\delta(\sigma(\vec{i}),(\sigma(\vec{j}))) \\ &+\sum_{\sigma} \lambda_{vol}(\tau)v(\sigma) - V_{target}(\tau(\sigma)))^2\end{aligned} \tag{1.1}$$

This hamiltonian, describes first the adhesion energy between different cell types. Therefor, every cell has a specific cell type $\tau(\sigma)$ [13], [14]. Each cell is placed onto a lattice with a spin $(\sigma(\vec{i},\vec{i})$ for every given dimension [12]–[14]. The adhesion energy between cells is only considered if the kroenecker delta is 0. Thus, the surface energy between cells is considered if $\delta(\sigma,\sigma') = 0$ [12]–[17]. Second, the volume of each cell is now considered and the user is allowed to add a multiplier to the hamiltonian, which is done by the multiplier λ_{vol}.

Together with Hogeweg they further develop their created extension of the CPM. The further developed model is today called GGH model. The main extension was that the user of GGH model is now able to add surface area constraints [12]–[14] as well as to use a negative boundary energy [12]. With the surface are constraint the formula for the GGH model is:

$$\begin{aligned}\mathcal{H}_{GGH} = &\sum_{\vec{i},\vec{j}} J(\tau(\sigma(\vec{i})),(\tau(\sigma(\vec{j})))(1-\delta(\sigma(\vec{i}),(\sigma(\vec{j}))) \\ &+\sum_{\sigma} \lambda_{vol}(\tau)v(\sigma) - V_{target}(\tau(\sigma)))^2 \\ &+\sum_{\sigma} \lambda_{sur}(\tau)s(\sigma) - S_{target}(\tau(\sigma)))^2\end{aligned} \tag{1.2}$$

The new model also allows the user to model (a): cell growth and proliferation (b): mitosis, i.e. cell division (c): fields, forces and diffusion and (d): chemotaxis and haptotaxis [12].

Glazier et. al. describe their model as:

> GGH models define biological structure consisting of the configuration of a set of *generalized cells*, each represented on a *cell lattice* as a domain of latitice sites sharing the same cell index [...], a set of *internal cell states* for each cell [...], and a set of *auxiliary fields* ..." [12].

The GGH model has the advantage that "Initial conditions emulating a particular biological configuration rather than random initial conditions." [12] and it has now biologically motivated properties instead of physically motivated properties [12].

1.2.3. CompuCell3D

CompuCell3D (CC3D) is an open-source program, which provides a simulation environment for multi- or single-cell-based modeling of tissues, organs and organisms [18]. To do so, CC3D uses the CPM in its simulation [18], which describes cell and Extended Cellular Matrix (ECM) behavior [19]. CC3D provides the possibility to create scripts for the simulation, e.g. cell growth, mitosis, apoptosis or necrosis scripts, in python, C++ or in their own CC3DM, which is their own Markup Language. With such scripts CC3D allows the user to modify the behavior of the simulation for a specific purpose. CC3D uses the GGH approach, explained in section 1.2.2 at page 3. It allows the user to choose between several cell-lattice types, i.e. a presentation of the pixels of a cell at a specific position in the simulation field (see picture XY). By default, it uses a square-lattice of single pixels for each dimension. There is also the possibility to use a hexagonal-lattice, where the pixels would be hexagons in two dimensions, or rhombic dodecahedrons in three dimensions. Since the core of a GGH simulation is the effective energy **IntroCC3D** CC3D tries to minimize this effective energy every Monte Carlo Step (MCS), i.e. a calculation step in the simulation. The basic form for the effective energy is:

$$\mathcal{H}_{boundary} = \sum_{\vec{i},\vec{j}} J(\tau(\sigma(\vec{i})), (\tau(\sigma(\vec{j})))(1 - \delta(\sigma(\vec{i}), (\sigma(\vec{j}))) \tag{1.3}$$

There are several ways to extend this form, it is possible to add a volume or a surface constraint. During each MCS an index-copy attempt takes place [20]. I.e. a pixel is selected, and it will be tried to overwrite a randomly chosen pixel, next to the current pixel, in order to minimize the effective energy. The index copy attempt succeed and takes place only if this index copy attempt decreases the effective energy **IntroCC3D** Each MCS the program tries to minimize the simulation, e.g.

with index copy attempts. The user has the chance to use her/his specific script every MCS, before the simulation starts or at the end of the simulation.

1.3. Objective of the Bachelor Thesis

The aim of this bachelor thesis is to create a 3D morphogenesis simulation of the urothelium using CC3D. Since the simulation models and python program for a 2D simulation are given and the simulation is done by CC3D the task is to modify the current application, of the 2D simulation, in a way that this program can be used for a 3D simulation of the different models.
Therefore, some parts of the program have to be modified. Some functionalities has to be written completely new wheras for other functionalities it is enough to modify these.
The result of this bachelor thesis will be presented with an of the 2D simulation realistic model. This model is choosen from the given models in the 2D simulation. The question how much more time the 3D simulation need than the 2D simulation will be covered in this thesis as well. If it is possible to provide a calculation of how much more effort a simulation in three dimensions need it will be included, otherwise an estimation of the more effort is presented.

1.4. Outline

In this chapter the knowledge to understand this bachelor thesis is provided. The next chapter provides the status at which the project was at the beginning of this bachelor thesis. Once the basic knowledge and the state of the art are explained, necessary modifications of the program are revealed. After these modifications are presented, the result of this bachelor thesis is shown. In the last section of this scientific work this thesis is discussed from different points of view and the conclusion out of this work is presented.

Chapter 2

State of the Art

The current moduro project has a stable 2D simulation of 16 different models using CC3D. With these models it is tried to make predictions about how bladder cancer arises. The simulations are performed by CC3D and then several values, e.g. the fitness of the model or how realistic the model is, etc., are summarized and displayed by the 'Moduro-Toolbox'. The project consits of several models and a program, both are written in python, i.e. a programming language. The models include properties of the specific model, e.g. adhesion energy or the possibility of the new cell types after mitosis, i.e. cell division, and the application modify the cell behavior, e.g. they check when a mitosis takes place, how fast the cell will growth, etc.. The program also calculates if the current model is a realistic one or not. These evaluating data are later used by the 'Moduro-Toolbox' to provide a overview over these data.

2.1. Display and Simulation of the Urothelium

In CC3D we simulate an urothel of the size of $333\,\mu\mathrm{m}$ for the x-axis and $100\,\mu\mathrm{m}$ for the y-axis. Because, CC3D has its constraints about the size if we use one core of the prozessor the size of the simulation will be rescaled to its maximum limitations of $267\,\mu\mathrm{m}$ for the x-axis and $80\,\mu\mathrm{m}$ for the y-axis in two dimensions. We use only one core for the simulation, because otherwise CC3D splits the simulation field into different grids and race conditions can occur at the edges of these grids [20]. I.e. a cell is in both grids, therefor one half is in one grid and the other half is in the other grid, during the simulation both grids calculate the volume of the cell and

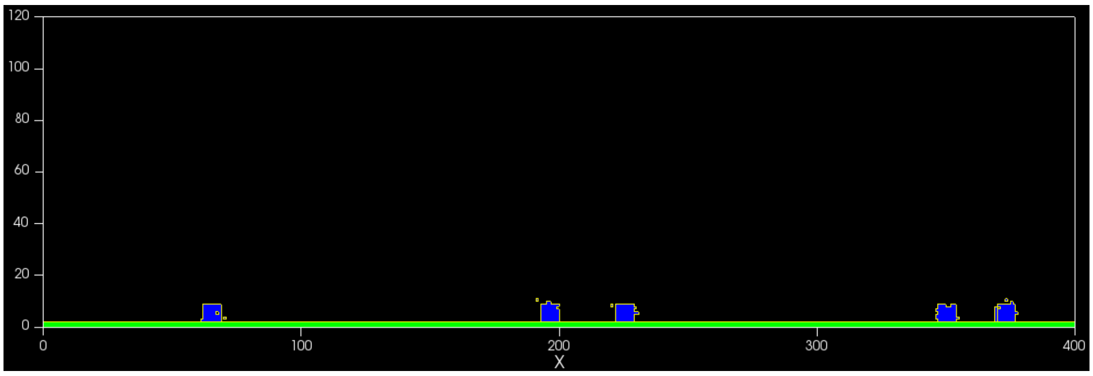

Figure 2.1.: Initial state of an 2D simulation of the model SpaPcdbCdiIn

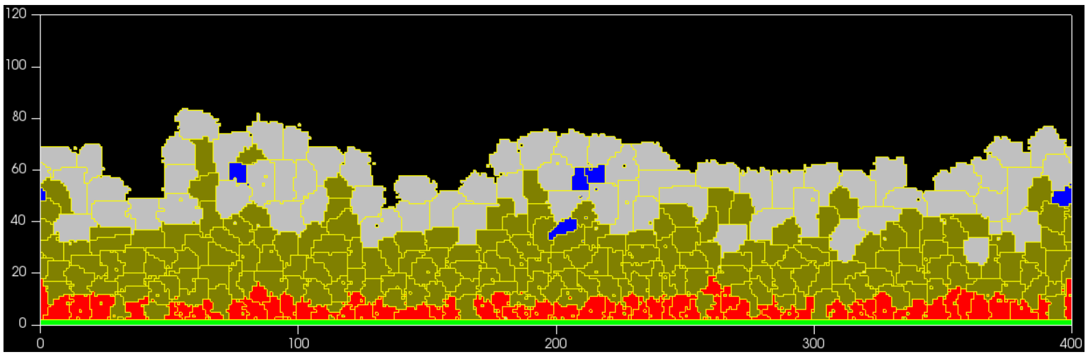

Figure 2.2.: 2D Simulation of the model SpaPcdbCdiIn after 33 days

both will apply the new volume of their calculation but the user do not know which calculation is correct. Thus, in order to reduce wrong results it is necessary to avoid race conditions.

The simulation of the urothel covers 720 days. At the first calculation step, MCS 0, the simulation is initialized, i.e. the cells are drawn and placed on the basal membrane. A illustration of an initial simulation is displayed in figure 2.1 at page 8. Since we simulate morphogenesis, the urothelium is proposed to growth and to proliferate in the given area. An illustration therefor is presented in figure 2.2 at page 8.

In order to provide a realistic simulation, the simulation has to be not to easy and not to hard, we have to simulate events which occur every MCS and some events which are occur not every MCS. The events, which are performed every MCS are a) cell growth and b) the check for mitosi c) cell death d) cell transformation and e) cell mutation. The event which does not occur every calculation step is f) urination. These events are presented in detail in the following subsections.

2.1.1. Events in the simulation

In this subsection the events for the simulation are presented. All of these events are executed every MCS, excluding the initializing step, MCS 0, and every factor of 250, MCS 250, 500, 750, etc..
The urination takes place every 12 hours, it should be simulated every 6 hours but because these events are not called every 250 MCS, it is only called every 12 hours.

Cell Growth

Every MCS we calculate the growth of a cell. In the project the maximum possible growth of a cell is calculated and applied. We need this calculation for the relation volume and target volume as well as for the relation volume, surface and target surface. Since, the volume and the surface of a cell is calculated by CC3D. The only way to influence it, is by setting the correct λ values, for the volume and surface, and by additional calculating the target volume and target surface values. In the program the calculation of the target volume and the target surface, i.e. the volume and surface which the cell should have, is made. CC3D uses the lambda mulitplier and the target values to calculate the effective energy. Of this calculated energy it is able to calculate the volume and the surface of each cell. This is why we calculate the target volume and surface instead of the actual volume and surface of a cell.

Mitosis

The verification if a cell divides or growth further is done in every simulation step. Doing so allows us to define a very specific MCS at which a cell divides, as there a 500 MCS per day. In order that a cell divides it needs to growth over its maximal size before it divides. The cells which are able to growth and as a result divide are a) stem cells b) basal cells and c)intermediate cells.

The umbrella cells are a product of mitosis than of cell division, if they would divide there would be two instead of one umbrella cell.

Necrosis

If a cell dies necrosis takes place. In the program a flag in the cell dictionary is set. Every MCS, the program checks if a cell dies or not. If so, the cell will shrink and as a result dissappear.

Mutation

After 2 days, 1000MCS, it is possible that a cell mutates, i.e. it becomes evil. We simulate the possibility for cells to mutate after 2 days, because otherwise there would not be much cells around the mutated cell. Each cell type has its own propability to become evil, in the current simulation it is not considered that cells mutate since the propability for each cell type to mutate is 0%.
If a cell mutates, there is also the flag for necrosis set. Thus, the cell shrinks and dissappears.

Transformation

A transformation can take place if a basal cell divides into a basal and a intermediate cell. Because the intermediate cell has to be inside the strata, it immediately will be transformed to an umbrella cell, which are kind the barrier to the urin in the bladder.

Urination

Every 12 hours an urination takes place. This is simulated in a way that randomly 2% of the cells in direct contact with the bladder are washed out [21]. In the program a flag for necrosis is set and the cells will dissappear because of the necrosis event.

2.2. Models

The 2D simulation of the urothel provided 16 different models. These model differ mostly in which cell types after the mitosis are created. The project divides the models into two domains, one has the id ’SSD’, which stands for "stem cell-like division" [21] and means that every time a stem cell divides there will be one stem

and one basal cell. The second domain has the id 'SPA', which stands for "stem cell population asymmetry" [21]. In the second domain the stem cell has a propability of 90% that it will be one stem and one basal cells after mitosis. There is also the chance with a probability of 5% that after mitosis of a stem cell there are two stem cells or two basal cells.
For the mitosis of basal cells there are 4 different models. The first one, 'BSD', describes the each basal cell which undergoes mitosis will create one basal and one intermediate cell. The second model describes that there is a 5% chance that the basal cell will become two basal or two intermediate cells. There is a 90% propability that the cell become one basal and one intermediate cell. The third model 'BPCD', describes that always a basal cells becomes two basal cells during mitosis and if the basal cell is not on the basal membrane it transform into an intermediate cell. The last model of the basal cells is the 'BCD'. In this model the basal cells immediately transform into an intermediate cell if it is not at the basal membrane anymore.
There are two different models how intermediate cell divide. One is the 'IPCD', a intermediate cell becomes always into two intermediate cells during mitosis and if there is no cell around the specific cell it will be transformed into an umbrella cell. The second one is the 'ICD', in this one only transformation of the intermediate cells into the umbrella cells happens.
With these different proliferation concepts there are 16 models in the project overall. These models were created by an earlier version of the project [21] and are displayed in table 2.1 at page 12.

2.3. Adhesion and cell sorting

During morphogensis of a strata the cells not only growth, they also sort themself. In order for the cell sorting there have to be different adhesion values **REF** i.e. how strong the surface of two different cell types are holding together. In the project this is done with a matrix, which every of the 16 model has. Such a matrix was created in an earlier version of the project [21] and is displayed in table 2.2 at page 13. In this table small values refer to more adhesion wheras larger values refer to less adhesion. The adhesion is calculated by CC3D. The cell type 'Medium' is a CC3D specific cell type and describe the space where no cells are in the simulation.

Type	ID	Description	Model
Stem cells	SSD	Stem cell-like division	$\begin{tikzcd} & B \\ S \arrow[ur] \arrow[dr] & \\ & S \end{tikzcd}$
	SPA	Stem cell population asymmetry	$\begin{tikzcd} & S \\ S \arrow[ur] \arrow[dr] & \\ & S \end{tikzcd}$ $p_s = 0.05$ $\begin{tikzcd} & B \\ S \arrow[ur] \arrow[dr] & \\ & S \end{tikzcd}$ $p_a = 0.90$ $\begin{tikzcd} & B \\ S \arrow[ur] \arrow[dr] & \\ & B \end{tikzcd}$ $p_s = 0.05$
Basal cells	BSD	Stem cell-like division in basal cell	$\begin{tikzcd} & B \\ B \arrow[ur] \arrow[dr] & \\ & I \end{tikzcd}$
	BPA	Basal cell population asymmetry	$\begin{tikzcd} & B \\ B \arrow[ur] \arrow[dr] & \\ & B \end{tikzcd}$ $p_s = 0.05$ $\begin{tikzcd} & I \\ B \arrow[ur] \arrow[dr] & \\ & B \end{tikzcd}$ $p_a = 0.90$ $\begin{tikzcd} & I \\ B \arrow[ur] \arrow[dr] & \\ & I \end{tikzcd}$ $p_s = 0.05$
	BPCD	Proliferation and contact differentiation of basal cells	$\begin{tikzcd} & B \\ B \arrow[ur] \arrow[dr] & \\ & B \end{tikzcd}$ and $\begin{tikzcd} B \arrow[r, dashed, "\neg \mathrm{BM}"] & I \end{tikzcd}$
	BCD	Only contact differentiation of basal cells	$\begin{tikzcd} B \arrow[r, dashed, "\neg \mathrm{BM}"] & I \end{tikzcd}$
Intermediate cells	IPCD	Proliferation and contact differentiation of intermediate cells	$\begin{tikzcd} & I \\ I \arrow[ur] \arrow[dr] & \\ & I \end{tikzcd}$ and $\begin{tikzcd} I \arrow[r, dashed, "\mathrm{M}"] & U \end{tikzcd}$
	ICD	Only contact differentiation of intermediate cells	$\begin{tikzcd} I \arrow[r, dashed, "\mathrm{M}"] & U \end{tikzcd}$

Table 2.1.: 16 different models derived in the project as there are different ways of proliferation and mitosis are simulated for the different cell types.

Types		M	BM	S	B	I	U
Medium	M	0	14	14	14	14	4
Basal membrane	BM		-1	1	3	12	12
Stem cell	S			6	4	8	14
Basal cell	B				5	8	12
Intermediate cell	I					6	4
Umbrella cell	U						2

Table 2.2.: Adhesion matrix for a model in the simulation. Smaller values refer to more adhesion and higher values mean less adhesion. The cell type M is the medium cell type, it is by CC3D a specific cell type which is every in the available space in the simulation, and BM is the basal membrane.

Cell type		V_{min}	d_{min}	V_{max}	d_{max}	Volume	Surface
Stem	S	268	8	523	10	perfect	average
Basal	B	381	9	523	10	important	average
Intermediate	I	905	12	1767	15	important	poor
Umbrella	U	1767	15	3591	19	important	poor

Table 2.3.: Constraints of a cell. Volumes V in μm^3, diameters d in μm. The 'Volume' and 'Surface' column describe how the λ_{vol} and the λ_{sur} should be set for each cell type.

2.4. Cell Properties

Each cell of cell type has several attributes, moreover each cell has an cell dictionary, in which additional attributes are stored. The properties regarding the cell type are likely what every cell in general has, e.g. min- max Diameter, min- max Volume, growth in µm per day or time until apoptosis, i.e. cell death, etc.. **Some of the properties are displayed in figure XY**. The attributes in the cell dictionary are more likely for the simulation. Therefore, these are some attributes which we are using to make decisions, e.g. the current and expected live time, a flag for necrosis can be set here, etc..

In the simulation we also need to know the physiology constraints of a cell. In an earlier version of the project these were evidenced and are displayed in table 2.3 at page 13 [21]. This table provides also the data how important the volume or the surface of a cell is.

2.5. Fitness functions

In order to validate the simulated models, there are several functionalities which check if the model is realistic or not. These functions, i.e. part of a program, check every day, every 500, MCS if the model is realistic or not [21]. The result of these two fitness functions is written into a file, and later read out by the moduro toolbox.

2.5.1. Arrangement fitness function

The arrangement fitness function ensure that the strata of the simulated urothelium has the correct order [21], i.e. that the first layer on the basal membrane consits only of stem and basal cells **REFS** the next three to five layers consits only of intermediate cells **REFS** and that there is one layer of umbrella cells **REFS**

$$f_a^* = \begin{cases} \dfrac{1}{(1-L_B)+(lib-L_I)+(1-L_U)+1} & \text{if amount of layers > 0} \\ 0 & \text{else 0} \end{cases} \tag{2.1}$$

In this equation L_B and L_U are boolean values, i.e. they have the value 0 or 1 [21]. They are 1 if the the first layer of cells consits only of basal or stem cells and if the most upper layer consits only of umbrella cells, otherwise they will be 0 [21]. lib is the amount of layers in betwenn the first and the last layers [21]. L_I contains the amount of layers, which consits only intermediate cells [21]. Therfore, $lib - L_I$ describes the amount cells which are not in there intended layer [21].
The arrangement fitness function is calculated columnwise, every $25\,\mu\text{m}$. After this calculation the average of all calculations of this function is calculated [21].

2.5.2. Volume fitness function

This function calculates the relative volume regarding the current volume of the different cell types in the urothel. The relative amount of the different cell types should be: stem and basal cell = 10%, intermediate cells = 67% and umbrella cells = 23% considering an average thickness of $85\,\mu\mathrm{m}$ [21]. Therfor the formula is:

$$f_{V_i} = \frac{1}{4(\frac{V_{Si} - V_{Ii}}{V_{Si}})^2 + 1} \tag{2.2}$$

V_{Si} and V_{Ii} describes the *should* and the actual *is* volume of a specific cell type i [21].

2.5.3. Overall fitness function

The overall fitness function calculates the total fitness out of the volume and the arrangement fitness function. Therefor the average of both functions is calculated [21]. The function is the following:

$$f(t_i) = \frac{f_V(t_i) + f_a^*(t_i)}{2} \tag{2.3}$$

t_i describes a specific time point, in MCS, at which this calculation could be done. At the end of the simulation is calculated as the average of the overall fitness function [21]. Therefor, the formula is:

$$f = \frac{1}{e+1} + \sum_{i=0}^{e} f(t_i) \tag{2.4}$$

$e + 1$ indicating the amount of calculations of the overall fitness function [21].

2.5.4. Moduro Toolbox

The purpose of the moduro toolbox is that we are able to evaluate a simulation. The toolbox itself was an bachelor thesis. All the data, which are written in to a file, e.g. cell birth and death by mitosis, data about the volume and arrangement fitness, etc., can be read out and analyzed with the moduro toolbox. It is possible to create a short video out of the simulation screenshots. This video is able to display the complete simulation within a few minutes. Therefor a extra program is required.

2.6. Simulation Functionalities

In order to not overload this chapter by presenting all functionalities, parts of the application which are important for the project and which were not modified in this bachelor thesis are presented. **A complete overview of the scripts is maybe displayed at figure XY.**

2.6.1. Position of the stem cells

After the amount of stem cells on the basal membrane is calculated, the cells will be positioned randomly on it. To do so a random value for the x and z Position is calculated in a way, that the cell will not be at the edge of the lattice. If the stem cell would be close to the lattice, than the proliferation could not take place in an optimal way.

```
for s in range(1, noStemCells + 1, 1):
    xPos = random.uniform(cellDiameter, self.execConfig.xLength -
        cellDiameter)
    zPos = random.uniform(cellDiameter, self.execConfig.zLength -
        cellDiameter)
    if self.execConfig.dimensions == 2:
        self._addCubicCell(2, xPos, 2, 0, cellDiameter, cellDiameter, 0,
            steppable)
    else:
        self._addCubicCell(2, xPos, 2, zPos, cellDiameter, cellDiameter,
            cellDiameter, steppable)
```

Listing 2.1: stem cell position

2.6.2. MinMaxVolume

In order that we are able to simulate mitosis a minimum and a maximum value regarding cell size is necesseray. This is done by the 'MinMaxVolume' in the cell Dictionary. These is done with a one dimensional array with two values, as it is displayed in the listing below:

```
cellDict['min_max_volume'] = [self.execConfig.calcVoxelVolumeFromVolume(cellType.
    minVol),
                 self.execConfig.calcVoxelVolumeFromVolume(cellType.maxVol)]
```

Listing 2.2: MinMaxVolume

Every cell type has its own minimum and maximum volume [21]. Since these values are saved in µm, they have to be converted to the voxel unit. With these values, in voxel, it is possible to set boundaries for the specific cell, e.g. to calculate the volume when mitosis takes place or the maximal volume of a cell of a specific cell type.

2.6.3. Volume and TargetVolume

As explained earlier, in order that the simulation starts the effective energy is not allowed to be 0. As we are initializing every cell, we set the target volume of every cell to be 1 larger than the current volume. This has the effect, that the simulation starts. During the simulation we calculate the target volume out of the possible growth volume and the current volume.

```
cell.targetVolume = cell.volume + 1 # At the beginning, the target is the actual
    size.
# cell.targetVolume = cellDict['normal_volume'] # At the beginning, the target is
    the actual size.
cell.targetSurface = self.execConfig.calcVoxelSurfaceFromVoxelVolume(cell.
    targetVolume)
```

Listing 2.3: set target Volume and Surface of a cell

In the same context we calculated the target surface as well. This has the same reason as for the target volume.

Chapter 3

Method

3.1. Sphere Cells

Since the program so far draws cubic cells and overgives the same parameter three times, a modification for this function was necessary. In the following the function call and the method of the old cell drawing method are displayed.

```
self._addCubicCell(2, xPos, 2, zPos, cellDiameter, cellDiameter, cellDiameter,
    steppable)
```

Listing 3.1: function call of the cell drawing method addCubicCell for a 3D cell

```
def _addCubicCell(self, typename, xPos, yPos, zPos, xLength, yLength, zLength,
    steppable):
    cell = steppable.newCell(typename)
    xPosDim = self.execConfig.calcPixelFromMuMeter(xPos)
    yPosDim = self.execConfig.calcPixelFromMuMeter(yPos)
    zPosDim = self.execConfig.calcPixelFromMuMeter(zPos)
    xLengthDim = self.execConfig.calcPixelFromMuMeter(xLength)
    yLengthDim = self.execConfig.calcPixelFromMuMeter(yLength)
    zLengthDim = self.execConfig.calcPixelFromMuMeter(zLength)
    # size of cell will be SIZExSIZEx1
    steppable.cellField[xPosDim:xPosDim + xLengthDim - 1,
              yPosDim:yPosDim + yLengthDim - 1,
              zPosDim:zPosDim + zLengthDim - 1] = cell
```

Listing 3.2: method to draw cubic cells

In the function 'addCubicCell' the parameters are converted into pixels, using the 'calcPixelFromMuMeter' function and then the cube will be drawn. To provide a deeper understanding of the of CC3D provided method, to draw the cell, the boundaries for the cube are displayed more generell in the following listing:

```
steppable.cellField[xStart:xEnd,
          yStart:yEnd,
```

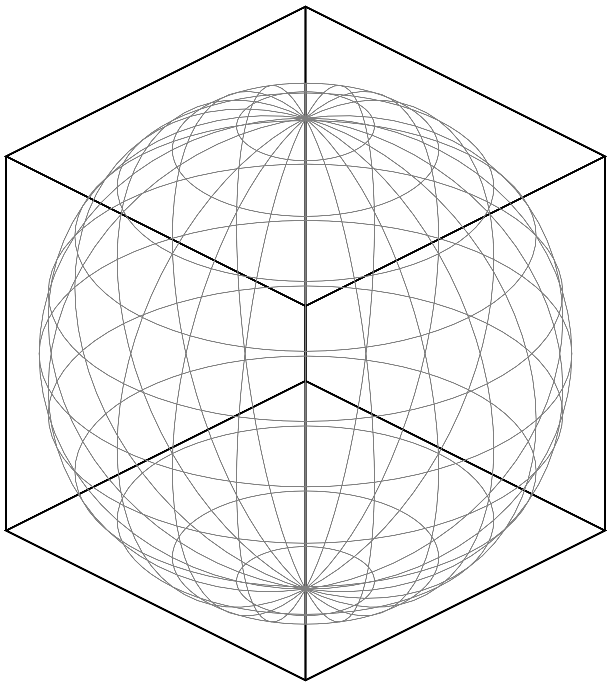

Figure 3.1.: A cuboid layed around a sphere

```
zStart:zEnd] = cell
```

Listing 3.3: boundaries of a drawn cuboid

In the listing above x,y,zStart defines the start and x,y,zEnd the end point of the to drwan area for each axis.
To be able to draw a cell as a sphere, CC3D has to allow the user to draw several the different voxels in the simulation field, all containing to one cell. Since CC3D allows the user to draw several pixels containing to one cell, a solution for this problem is possible.

In order to draw a sphere out of voxels, a cuboid has to lay around the sphere, as it is displayed in 3.1 at page 20. The cuboid has to be at least as large as $2 \cdot radius$ of the sphere, as it is displayed figure 3.2 at page 22. The cuboid is filled with voxels. To be able to calculate every point within the sphere, the cuboid and the sphere are required to have the same center **REF** In this case the following equation provides a mechnanism in which every point within the sphere can be calculated.

$$\sqrt{(x_r - x_0)^2 + (y_r - y_0)^2 + (z_r - z_0)^2} <= radius \qquad (3.1)$$

In the equation x_r, y_r and z_r describe the current point of each of the three axis and x_0, y_0 and z_0 describe the center of the sphere. If the distance of the current x, y, z point is smaller or equal to the radius the current point is within the sphere otherwise it would be outside of the sphere. This equation has the weakness that only one point is considered. Thus, only one point of the voxel is considered in the decision if the complete voxel is within the sphere or not. Since every voxel is a cuboid itself, the length of all corners are the same. This fact can be used to increase the accuracity of the equation above. Because we are able to calculate the corner length of a voxel, we are able to decide if the center and not one point of a voxel is at the inside or outside of a sphere. Because, the radius of the sphere is known, therefore the diameter of the cuboid can be calculated, and the voxel density is also known, the corner length of each voxel can be calulated as it is displayed in the following equation.

$$c = \frac{2 \cdot radius}{voxeldensity} \tag{3.2}$$

In this equation c describes the corner length of a voxel. With this corner length it is now possible to calculate the center of a voxel and use this center further to decide if the voxel is inside or outside of the sphere. To receive the center of a voxel the calculation has to be done for every of the three axis. The following three formulas in equation **xy** displays such a calculation.

$$\begin{aligned} x_c &= x_r + \frac{x_{r+c} - x_r}{2} \\ y_c &= y_r + \frac{y_{r+c} - x_y}{2} \\ z_c &= z_r + \frac{z_{r+c} - x_z}{2} \end{aligned} \tag{3.3}$$

In this equation x_r, y_r and z_r describe the start point and x_{r+c}, y_{r+c} and z_{r+c} describe the end point of the voxel. With the calculations of equation **xy** it is now possible to considere the center of a voxel in the decision if the voxel is in- or outside the sphere, as it is displayed in equation **xy**.

$$\sqrt{((x_c - x_0)^2 + (y_c - y_0)^2 + (z_c - z_0)^2} <= radius \tag{3.4}$$

As long as both, the cuboid and the sphere, have the same center it is possible to calculate every point inside or outside the sphere**REF** To do so, for every point within the cuboid it is necessary to calculate if the distance to the center is smaller

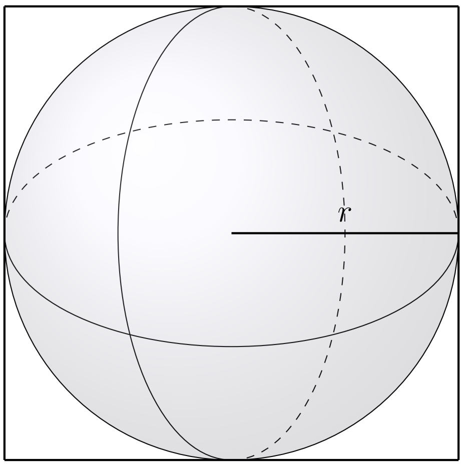

Figure 3.2.: A cuboid with minimal size layed around a sphere

or equal to the radius of the sphere. If this is the case, the point is inside of the sphere, otherwise it is outside of the sphere. Thus, the formula to determine if a point is inside of the sphere is the following:

$$\sqrt{(x_r - x_0)^2 + (y_r - y_0)^2 + (z_r - z_0)^2} <= radius \tag{3.5}$$

In this formula x_r, y_r, z_r contains the current point of the specific axis and x_0, y_0, z_0 is the center of the cuboid and the sphere. Whenever the distance of the current point in the cuboid to the center is smaller or equal the radius of the sphere, than this point is inside of the sphere.
In order that it is possible to draw the cells in this way, the iteration over all points of the cuboid is necessary. Because to do this is computational more expensive than to just draw a cube, the cuboid should be as small as possible. Still, the cuboid has to be at least as large as $2 * radius$ of the sphere, as it is displayed figure 3.2 at page 22.

Since the old method is required to draw the basal membrane throughout the whole area, a new function is needed. The old function is refactored into 'addMembrane', i.e. renamed and every time the function name appears within the project, within a comment or the function is called, this appearance is also change. With the new method name it clear that this function is now, in the 3D simulation, only required for the basal membrane, since it is a cubic cell, and if some simulations in two dimensions are made.
The created function 'add3DCell', it is displayed in the listing below, is similiar to

the old one, as it also has to convert the used points, i.e. the start and end point of the cuboid and the center of the cuboid and the sphere, into pixels. Then the iteration of all three axis as well as the calculation to decide if the pixel is inside the sphere or not takes place.

```
def _add3DCell(self, typename, xPos, yPos, zPos, radius, steppable):
    '''The parameters are all in micro meter
    wheras the calculated variables are in px'''

    print 'x:{}, y:{}, z:{} r:{}'.format(xPos, yPos, zPos, radius)
    cell = steppable.newCell(typename)
    xStart = self.execConfig.calcPixelFromMuMeter(xPos - radius)
    x0 = self.execConfig.calcPixelFromMuMeter(xPos)
    xEnd = self.execConfig.calcPixelFromMuMeter(xPos + radius)
    yStart = self.execConfig.calcPixelFromMuMeter(yPos - radius)
    y0 = self.execConfig.calcPixelFromMuMeter(yPos)
    yEnd = self.execConfig.calcPixelFromMuMeter(yPos + radius)
    zStart = self.execConfig.calcPixelFromMuMeter(zPos - radius)
    z0 = self.execConfig.calcPixelFromMuMeter(zPos)
    zEnd = self.execConfig.calcPixelFromMuMeter(zPos + radius)

    radiusPx = self.execConfig.calcFloatPixel(radius)
    stepLength = 1.0
    print 'steplength {}, zStart + stepLength/2 {}'.format(stepLength, (zStart
        +(((zStart+stepLength) - zStart)/2.)))
    print 'x:{}-{}, y:{}-{}, z:{}-{} radiusPx:{}'.format(xStart, xEnd, yStart,
        yEnd, zStart, zEnd, radiusPx)
    # loop over the center of each pixel to determine boundaries of the circle
    for xr in xrange(xStart, xEnd):
        for yr in xrange(yStart, yEnd):
            for zr in xrange(zStart, zEnd):
                rd = sqrt(
                    ((xr+(((xr+stepLength) - xr)/2.)) - x0) ** 2 +
                    ((yr+(((yr+stepLength) - yr)/2.)) - y0) ** 2 +
                    ((zr+(((zr+stepLength) - zr)/2.)) - z0) ** 2)
                if (rd <= radiusPx):
                    steppable.cellField[xr, yr, zr] = cell
```

Listing 3.4: created method to draw a spere cell

With this created function it is now possible to draw a cell as a sphere, as it is displayed at figure XY **XY**
To draw the cell as a sphere is the first step to have sphere cells in the simulation. Two main parts of the simulation are the growth and the mitosis of the simulation. In order that the cells are able to stay as a sphere it is required to adjust the volume and surface calculation as well.

3.2. Lambda Multiplier

In the project there were several places, where a lambda multiplier for the effective energy, see section 1.X, is calculated, but it is only set once. There are two additional multipliers, for the surface and volume constraints, saved in within each cell object. Since we are not using these values in the programm and they do not influence or set the λ_{vol} or λ_{sur} which CC3D uses, these two values were outcommented in the first step and later deleted.
Also in one place there were methods to calculate the lambda values. Since this methods are not used in the project and moreover they had a multiplier itself to calculate the multiplier for the specific effective energy, as it is displayed in the following listing, these were also first outcomented and later deleted.

```
def calcSurLambdaFromSurFit(self, surFit):
    return 100.0 * surFit

def calcVolLambdaFromVolFit(self, volFit):
    return 1.0 * volFit
```

Listing 3.5: methods to calculate the λ_{vol} and λ_{sur} for the effective energy

In the project the multipliers for the effective energy are now set each time when a cell is initialized. A second time the multiplier λ_{vol} is set, is during necrosis, in generell if a cell dies. Thus, there is now the advantage that at a specific point in the coding these values are set.

3.3. Approximation Error

Since the project includes conversions from µm to pixels and the amount of pixels, e.g. for the surface of a cell, have to be set in integer, i.e. a whole number, it is possible that in some places in the program there are approximation errors. In the project, the values of µm are saved either with the data type double or float – both allow several decimal digits. To set these values as a whole number the values are casted, i.e. the datatype of a variable, i.e. a placeholder, will be changed – in this context the values of type double or float will be transformed to a whole number by cutting of the decimal digits. At some points in the project the values of µm are converted to a whole number and then it is further calculated with this casted value. To cast the value should always be the last step in order to avoid approximation and calculation errors. In the following listings an example of to early casting is displayed:

```
def calcVoxelVolumeFromVolume(self, volume):

    r = (3 * volume / (4.0 * PI)) ** (1.0 / 3.0) # Radius of a sphere with
        known volume.
    rDimension = self.calcPixelFromMuMeter(r) # Convert it to a pixel unit.

    if self.dimensions == 2:
        return self.__truncate(PI * (rDimension ** 2)) # Area of a circle.
    else:
  return self.__truncate(4.0 / 3.0 * PI * (rDimension ** 3)) # Volume of a
      sphere.
```

Listing 3.6: example of an calculation error because of an too early executed cast

```
def calcPixelFromMuMeter(self, mum):
return int(self.voxelDensity * mum + 0.000001)
```

Listing 3.7: function to convert values of µm into pixel

In this example the second command of the ’calcVoxelVolumeFromVolume’ function calls the function of the second listing, in which the approximation error happen. Since the decimal digits are just cut off, the further calculation contains a wrong value and as a consequence the result of the calculation is incorrect.
In order to solve the problem the complete calculation is calculated with decimal digits. After the calculation is done it will be checked if the first decimal digit is larger or equal to 5. If this condition is true the result is increased by 1 otherwise not, as a last step the result is casted. This technique has the advantage that calculation errors due to casting are decreased, because the cast is the very last step. It is possible that in the program still have some rounding errors, but these are not as dramatic as an casting error in the calculation. To remove this casting error, the function ’calcVoxelVolumeFromVolume’ is extended in the following way.

```
def calcVoxelVolumeFromVolume(self, volume):
    r = (3 * volume / (4.0 * PI)) ** (1.0 / 3.0) # Radius of a sphere with
        known volume.
    rDimension = r * self.voxelDensity
    if self.dimensions == 2:
        return int(self.__truncate(PI * (rDimension ** 2))) # Area of a circle.
    else:
        result = 4.0 / 3.0 * PI * (rDimension ** 3)
        if result % 1.0 >= 0.5:
            result += 1

        return int(result)
```

Listing 3.8: the same function as before but without approximation errors because the cast and the rounding is done after the calculation

One use of the function is to calculate the volume constraints of a specfic cell type. In the simulation a minimum and a maximum volume for each cell type is calculated by the function 'calcVoxelVolumeFromVolume'. These value are used in the calculation to determine if mitosis takes place or not.
For the basal cell the minimum volume is 381 µm and the maximal volume is 523 µm. With the old calculation the minimum volume constraint would be the same as for the stem cells –905vx. Since the current calculation only casts and round at the very last step, the result is now 1286vx, which is a more precise and correct result. In the following table 3.2 an example of to early rounding and casting is displayed:

Table 3.1.: Stakeholders' motivations to use **SW CS!**

Volume in µm	radius in µm	rounded or casted to	not rounded result in vx	rounded result in vx
381	4.497	not rounded or casted	1285.67	1286
381	4.497	4	904.779	905
381	4.497	5	1767.15	1767

This table provide three possible solutions to calculate the volume in voxel of a given volume in µm. The first row calculates the voxel volume with the modified function. The second row is the calculation of the program without the modification. Thus, the calculated radius is casted and then used in the further calculation. The third row provides a possible calculation in which the radius is rounded and then used for the further calculation.
A similiar calculation error is done in the calculation of the target surface of each cell. The calculation of the target surface is done once by initalizing the simulation and every MCS. This calculation is necessary that CC3D is able to calculate the effective energy and that it is able to determine how the cell should look like, what shape it should have.
Because, in the simulation the volume is calulated in every MCS, a calculation of the surface of a sphere with a given volume is more precise than the calculation with the radius. We use the of CC3D calulated volume to calculate the target surface. To

calculate the surface with a given volume the formula 3.6 has to be modified, as it is displayed in the formula 3.8 below:

$$Surface_{sphere} = 4 * \pi * r^2 \tag{3.6}$$

To calculate the radius out of a given volume of a sphere, the formula to calculate the volume of a sphere has to be shifted as it is displayed in formula 3.7.

$$\begin{aligned} Volume_{sphere} &= \frac{4}{3} * \pi * r^3 \\ 3 * Volume_{sphere} &= 4 * \pi * r^3 \\ r^3 &= \frac{3 * Volume_{sphere}}{4 * \pi} \\ r &= \sqrt[3]{\frac{3 * Volume_{sphere}}{4 * \pi}} \\ r &= (\frac{3 * Volume_{sphere}}{4 * \pi})^{\frac{1}{3}} \end{aligned} \tag{3.7}$$

If the formula 3.7 is inserted in the formula 3.6 the following formula to calculate the surface out of the volume of a sphere is the result:

$$\begin{aligned} Surface_{sphere} &= 4 * \pi * r^2 \\ &= 4 * \pi * (\frac{3 * Volume_{sphere}}{4 * \pi})^{(\frac{1}{3})^2} \\ &= 4 * \pi * (\frac{3 * Volume_{sphere}}{4 * \pi})^{\frac{2}{3}} \end{aligned} \tag{3.8}$$

The following listings displays how the target surface is calculated, with the formula of 3.8 in the program. To calculate the target surface, the current target volume of the cell is given as parameter 'voxelVolume'. The formula to calculate the surface of a sphere, with the radius, is:

```
def calcVoxelSurfaceFromVoxelVolume(self, voxelVolume):
    if self.dimensions == 2:
        # some fractal factor!
        return self.__truncate(1.5 * 2 * (PI * voxelVolume) ** (1.0 / 2.0)) #
            Circumference.
    else:
  return self.__truncate(2.0 * 4 * PI * (3 * voxelVolume / (4 * PI)) ** (2.0 /
     3.0)) # Surface.
```

Listing 3.9: calculation of the target surface of a cell

```
    def __truncate(self, value):
        res = int(value + 0.00001)
        if res <= 1:
            return 1 # Ensure that size is at least 1.
        else:
      return res
```

Listing 3.10: calculation of the target surface of a cell

In order that the cell will growth it is required to set a factor, which increases the calculated target surface of the cell. After the calculation of the target surface is done the function 'truncate' is called. This function simply casts the result of the calculation. Also, if the given parameter is smaller than 1 it will be increased to 1, this is for a different functionality of the program and will not be further noticed. To do cast as the last step is correct, but a rounding is required before. With this functions it is possible that a target surface of $623,873$ is set to 623 instead of to 624. There are several ways to solve this problem. I choosed to ..... because.......

3.4. Area of stem cells on the basal membrane

In earlier version of the project it was evidenced that around 12% of the area of the basal membrane are required to be filled with stem cells in order to have an optimal proliferation during the morphogenesis of the cells [21]. In the project the calculation of the amount of stem cells for two dimensions were correct but without an mathematical evidence, as it is shown in the following listing:

```
    def _initCells(self, steppable):
 ...
        cellDiameter = self.cellTypes[2].getAvgDiameter()
        stemCellFactor = 8 * cellDiameter

        if self.execConfig.dimensions == 2:
            noStemCells = int(self.execConfig.xLength / stemCellFactor)
        else:
            noStemCells = int(self.execConfig.xLength * self.execConfig.yLength /
                              (stemCellFactor * stemCellFactor))
 ...
```

Listing 3.11: calculation of the amount of stem cells on the basal membrane without a mathematical evidence

Since the y-axis is negligible in the calculation of an area, the calculation for two dimensions considers the x-axis and the calculation for three dimensions considers the x- and z-axis. Therfore, in two dimensions, only considering the x-axis, the area of the stem cells should be calculated by using the cell diameter. This situation is displayed in figure 3.3 at page 29. For three dimensions it is possible to calculate the area of a circle with the formula $\pi * radius^2$, since a circle is in two dimensions, and use this calculation to further determine the amount of stem cells on the basal membrane. An example for the basal membrane and a stem cell in three dimensions is displayed in figure 3.4 at page 29.

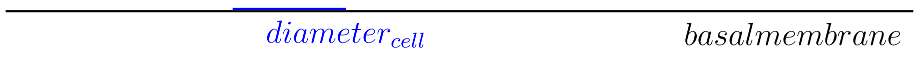

Figure 3.3.: considered area to spread the stem cells in two dimensions with an example of one stem cell placed on the basal membrane. Because only the x-axis is displayed we need to calculate the diameter of a cell

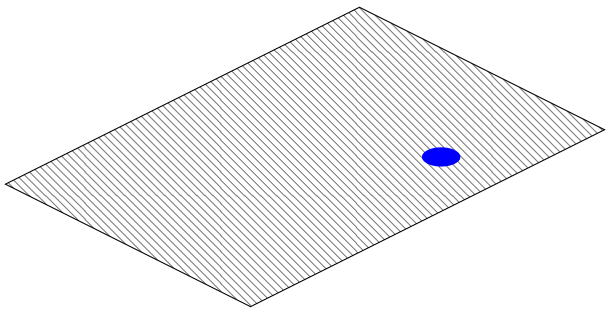

Figure 3.4.: considered area to spread the stem cells in three dimensions with an example of one stem cell placed on the basal membrane. This stem cell is a circle, because for the calculation we use two dimensions

The following formulas calculate the amount of stem cells in two dimensions:

$$A_{stemcells} = xLength * 0.12 \tag{3.9}$$

$$Amount_{StemCells} = \frac{A_{stemcells}}{diameter} \tag{3.10}$$

Formula 3.9 ensures that only 12% of the given area is used. Formula 3.10 then calculates the amount of stem cells on this given area. Since the result of the calculation is often not a whole number, the result is checked if the first decimal digit is larger or equal than 5 and then it is rounded up or down. This has a significant difference in the result, as it is displayed in table 3.2 at page 30.

As the table displays, it is important to round the result. Otherwise there would be an approximation error which is to avoid.
For three dimensions the formulas 3.9 and 3.10 have to be extended, because the

Table 3.2.: Possible approximation error by not rounding the result of formula 3.10

XLength in µm	Result of $Amount_{StemCells}$	rounded or casted result	Area used of Stem cells in µm	relative used area
200	~ 2.66	3	27	13.5%
200	~ 2.66	2	18	9%

z-axis is now also considered in the calculation of the amount of stem cells on the basal membrane. Therfore, the extended formulas for three dimensions are:

$$A_{stemcells} = xLength * zLength * 0.12 \tag{3.11}$$

$$Amount_{StemCells} = \frac{A_{stemcells}}{\pi * r^2} \tag{3.12}$$

The result of formula 3.12 has to be rounded as well, otherwise the program would again include rounding errors, as it is shown in table 3.2. That the formulas are correct is shown with the following example. In order to keep a good overview of the results and the parameters in the formulas, the decimal digits are rounded after the third decimal digit.
For a simulation with a length of $200\,\mu\text{m}$ at the x-axis and a length of $50\,\mu\text{m}$ at the z-axis, the overall area is $10\,000\,\mu\text{m}^2$ and the for the stem cells reserved 12% of the membrane are $1200\,\mu\text{m}^2$. With an average diameter of $9\,\mu\text{m}$, the minimum and maximum diameter of the different cells are displayed in table **??** at page **??**, the result of the amount of stem cells is ~ 18.863 which is rounded up to 19 stem cells on an area of $10\,000\,\mu\text{m}^2$.

$$\begin{aligned} Amount_{StemCells} &= \frac{200\,\mu\text{m} * 50\,\mu\text{m} * 0.12}{\pi * r^2} \\ &= \frac{1200\,\mu\text{m}^2}{63.617\,\mu\text{m}^2} \\ &\sim 18.863 \\ &= 19 \end{aligned} \tag{3.13}$$

To validate this result, the area of 19 circles with a radius of $\frac{9\,\mu\text{m}}{2}$ has to be calculated. To then get the relative amount of stem cells on the basal membrane it is required to divide the result of $Amount_{StemCells}$ with the overall possible area of the urothel. As the calculations show, on an area with $10\,000\,\mu m^2$ to draw 19 cells uses 12% of the given area.

$$\begin{aligned} A_{StemCells} &= 19 * \pi * cellradius^2 \\ &= 19 * \pi * 4.5\,\mu m^2 \\ &\sim 1208.728\,\mu m^2 \end{aligned} \tag{3.14}$$

$$\begin{aligned} A_{StemCells} &= \frac{A_{StemCells}}{10\,000\,\mu m^2} \\ &= \frac{1208.728\,\mu m^2}{10\,000\,\mu m^2} \\ &= 0.120 \\ &= 12\% \end{aligned} \tag{3.15}$$

In the following listing the new calculation to figure out the amount of stem cells on the basal membrane at the start of the simulation is shown. The calculation is for two as well as for three dimensions modified, since both did not have a mathematical evidence.

```
    def _initCells(self, steppable):
...
        cellDiameter = self.cellTypes[2].getAvgDiameter() # cell diameter is of
            type float


        if self.execConfig.dimensions == 2:
            noStemCells = int(self.execConfig.xLength * 0.12 / cellDiameter)
        else:
            noStemCells = ((self.execConfig.xLength * self.execConfig.zLength) *
                0.12) / (PI * (cellDiameter / 2.) ** 2)

        if noStemCells % 1 > 0.5:
            noStemCells += 1

        noStemCells = int(noStemCells)
...
```

Listing 3.12: new calculation of the amount of stem cells on the basal membrane with a mathematical evidence

3.5. TargetVolume, Surface after Mitosis

Mmitosis is done by CC3D, i.e. CC3D decides where the cell splits and calculates the volumes and the surface of the two new cells. In our program we calculate and set attributes of the new cells, e.g. the target volume or the target surface. The target volume is simply calculated by dividing the target volume of the cell before mitosis by 2. This value is applied for both new created cells. The problem with this technique is that it is possible that the cell does not split complety in the middle. Therefore, it is not possible to set the target volume of the two created cells without the knowledge of the volume of both cells. Moreover, the function 'initCellAttributes' calles the function 'setCellAttributes', in which the targetVolume of the specific cell is set to be 1 larger than the current volume. If the commands of the function 'initCellAttributes' are complete and the program is again in the below listet function, then the target volume of both cells is set to a value which has no evidence to be correct.

```
def updateAttributes(self):
    parentCell = self.mitosisSteppable.parentCell
    childCell = self.mitosisSteppable.childCell

    #Has to be done this way otherwise if cell.volume is chosen than it
        disappears
    newVol = parentCell.targetVolume / 2

    descendents = self.model.cellTypes[parentCell.type].getDescendants()
    parentCell.type = descendents[0]
    childCell.type = descendents[1]

    # Now set the attributes for the two daughter cells:
    cellDictChild = self.getDictionaryAttribute(childCell)
    self.model.initCellAttributes(childCell, cellDictChild)
    cellDictParent = self.getDictionaryAttribute(parentCell)
    self.model.initCellAttributes(parentCell, cellDictParent)

    parentCell.targetVolume = newVol
    childCell.targetVolume = newVol

    # Register events
    self._cellLifeCycleBirth(parentCell)
    self._cellLifeCycleBirth(childCell)
cellDictChild['colony'] = cellDictParent['colony']
```

Listing 3.13: two cells are created by mitosis where the target volume is set at a wrong place

```
def setCellAttributes(self, cellDict, cell, lifeTimeParent):
...
    cell.targetVolume = cell.volume + 1 # At the beginning, the target is the
        actual size.
    cell.targetSurface = self.execConfig.calcVoxelSurfaceFromVoxelVolume(cell.
        targetVolume)
    ...
```

Listing 3.14: set the target volume as well as the target surface in the intended function

Since CC3D calculates the new volume of both created cells and the cell attributes are initialized and set in the functions 'initCellAttributes' and 'setCellAttributes', which is called by 'initCellAttributes', setting the target volume for both cells like it is in 3.5 is not correct. Thus, these commands are deleted and the target volume will now only be set at the function 'setCellAttributes'. There the target volume is 1 larger than the current volume, because the cells are just created. With the next MCS, and then every MCS, the target volume will be calculated regarding the growth per day of the specific cell type. Thus, no changes in the function 'setCellAttributes' were requirded, the actual code of the function 'updateAttributes' is the following:

```
def updateAttributes(self):
    print 'GrowthMitosisSteppable.updateAttributes()'
    parentCell = self.mitosisSteppable.parentCell
    childCell = self.mitosisSteppable.childCell

    # return cell types based on the probability (in ****Ua) two descendents of
         the current cell
    descendents = self.model.cellTypes[parentCell.type].getDescendants()
    parentCell.type = descendents[0]
    childCell.type = descendents[1]

    # Now set the attributes for the two daughter cells:
    cellDictChild = self.getDictionaryAttribute(childCell)
    self.model.initCellAttributes(childCell, cellDictChild)
    cellDictParent = self.getDictionaryAttribute(parentCell)
    self.model.initCellAttributes(parentCell, cellDictParent)

    print 'parentCell.volume {} >= parentCell.targetVolue {}'.format(parentCell
        .volume, parentCell.targetVolume)
    print 'childCell.volume {} >= childCell.targetVolue {}'.format(childCell.
        volume, childCell.targetVolume)

    # Register events
    self._cellLifeCycleBirth(parentCell)
    self._cellLifeCycleBirth(childCell)
    cellDictChild['colony'] = cellDictParent['colony']
```

Listing 3.15: two cells are created due to mitosis

Chapter 4

Results

ergebnisse (screenshots source code)

4.1. Draw Sphere Cells

With the created method the program is now able to draw a sphere. Since we use voxels in the 3D simulation, it is not possible to draw a round sphere. Thus, an approximation to the volume, surface and the shape has to be done with these voxels. This approximation to a real sphere allows us to draw cells, which look like spheres, and also let them growth with their shape. Since in the simulation are lattice constraints as well as other cells, it is possible that the shape of a sphere will not be kept during the simulation. Moreover, the deviation of the volume and surface between the in pixels drawn and grown sphere to a real sphere grows **expontially** as the radius of the sphere grows as it is displayed in **figure XY**. A result of a drawn cell with different radius and voxel densities is displayed at **figure xy**.

A factor the approximation to an real sphere is the voxel density. It describes how many voxel display $1\,\mu\mathrm{m}$. As an example, if the voxel density = 1 than it represents $1\,\mu\mathrm{m}$, if the voxel density = 2 than one voxel represents $0.5\,\mu\mathrm{m}$ and so on. In a simulation with $100\,\mu\mathrm{m}$ at the x-axis, $80\,\mu\mathrm{m}$ at the y-axis and $50\,\mu\mathrm{m}$ at the z-axis is displayed with 100 voxel at the x-axis, 80 voxel at the y-axis and 50 voxel at the z-axis for a voxel density of 1. With a voxel density of 2 the lattice of the simulation would include 200 voxel at the x-axis, 160 voxel at the y-axis and 100 voxel at the z-axis. Thus, the radius of a cell is also dependend of the voxel density. Considering that the radius growths as the voxel density growths and that between the calculated

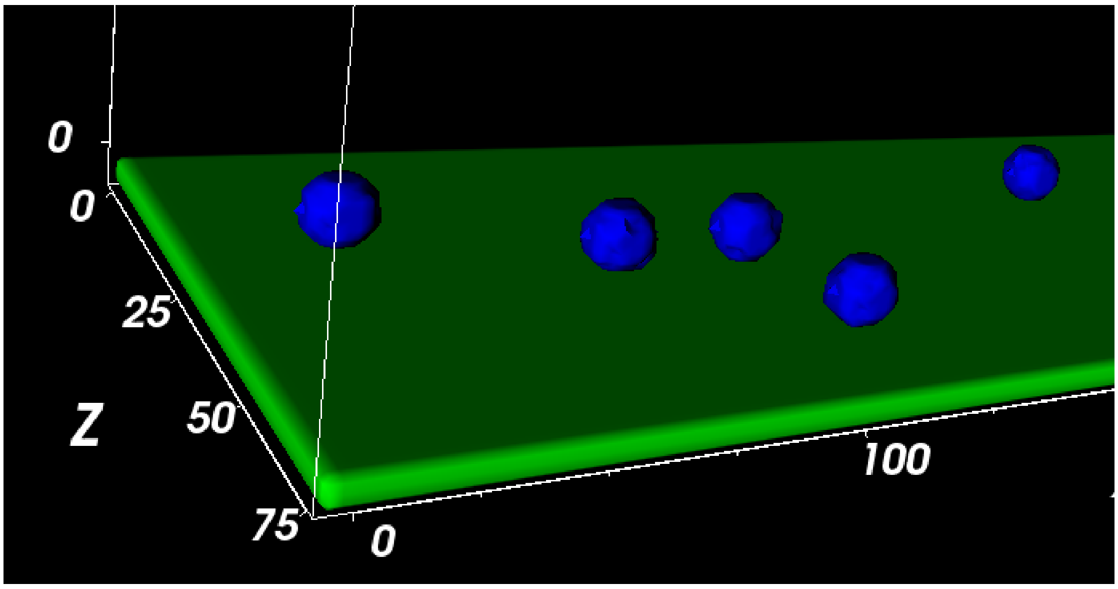

Figure 4.1.: A cell drawn as a sphere

sphere, out of pixels, and a real sphere the deviation growths further apart as the radius growths it is not correct to say that with a higher voxel density there will be a more accurate result. Because, the approximation error is larger for voxel density of 2 it is possible that the approximation errors effects the accuracity of the simulation in a negative way. This fact has to be checked in a future work.

Chapter 5

Conclusion

Final words

- take a look a biocellion (no race conditions) and morphois
- use chemical field for growth
- Random Walker algorithm
- kugel oberfläche mit pyramiden berechnen (problem wir haben immernoch pixel und square lattice)

  archimedes volumen eines kugelschnittes

5.1. Draw Sphere Cells

It is also possible to use only the effective energy

5.2. Calculate Surface Variation Pixels to Sphere

Two possibilities to design the algorithm

- check the center of each pixel for the radius, is within the radius or not
- calculate the y-values of the radius for an x-value of the center of several pixels -> decide dependen on the result of the y-values if the specific pixel is counted or not (if the center of the pixel is within or outside the radius)

5.3. lambda values

This was the only way to do it

5.4. Approximation & calculation Errors

Find a more elegant way to do it.
Floating point arithmetic

5.5. CC3D

- **CC3D sucks**
- 400vx · 400vx · 400vx = 107171875vx CC3D has its problem and crashes sometimes on 4GB RAM
- CC3D does not empty the used RAM after a simulation -> after several simulations the computer will run out of RAM an CC3D crashes
- no Debug -> testing with print commands at command line

List of Abbreviations

GGH	Glazier-Graner-Hogeweg
CC3D	CompuCell3D
ECM	Extended Cellular Matrix
MCS	Monte Carlo Step
CPM	Cellular Potts Model
EPM	Extended Potts Model

List of Tables

List of Figures

Listings

Bibliography

[1] N. J. Poplawski, U. Agero, J. S. Gens, M. Swat, J. A. Glazier, and A. R. A. Anderson, "Front instabilities and invasiveness of simulated avascular tumors", *Bulletin of Mathematical Biology*, vol. 71, no. 5, pp. 1189–1227, Jul. 2009. DOI: 10.1007/s11538-009-9399-5. [Online]. Available: https://doi.org/10.1007/s11538-009-9399-5.

[2] [Online]. Available: https://www.everydayhealth.com/bladder-cancer/guide/.

[3] [Online]. Available: https://www.cancer.org/cancer/bladder-cancer/about/what-is-bladder-cancer.html.

[4] M. Lazzeri, "The physiological function of the urothelium–more than a simple barrier", *Urologia internationalis*, vol. 76, no. 4, pp. 289–295, 2006. DOI: https://doi.org/10.1159/000092049.

[5] T. Yamany, J. Van Batavia, and C. Mendelsohn, "Formation and regeneration of the urothelium", *Curr Opin Organ Transplant*, vol. 19, no. 3, pp. 323–330, Jun. 2014. DOI: 10.1097/MOT.0000000000000084.

[6] L. A. Birder, "More than just a barrier: Urothelium as a drug target for urinary bladder pain", *American Journal of Physiology-Renal Physiology*, vol. 289, no. 3, F489–F495, 2005, PMID: 16093424. DOI: 10.1152/ajprenal.00467.2004. eprint: http://www.physiology.org/doi/pdf/10.1152/ajprenal.00467.2004. [Online]. Available: http://www.physiology.org/doi/abs/10.1152/ajprenal.00467.2004.

[7] A. A. Karl-Erik Andersson, "Urinary bladder contraction and relaxation: Physiology and pathophysiology", *Physiological Reviews*, vol. 84, no. 3, pp. 935–986, 2004, PMID: 15269341. DOI: 10.1152/physrev.00038.2003. eprint: http://www.physiology.org/doi/pdf/10.1152/physrev.00038.2003. [Online]. Available: http://www.physiology.org/doi/abs/10.1152/physrev.00038.2003.

[8] G. Apodaca, "The uroepithelium: Not just a passive barrier", *Traffic*, vol. 5, no. 3, pp. 117–128, 2004. DOI: 10.1046/j.1600-0854.2003.00156.x. [Online]. Available: http://dx.doi.org/10.1046/j.1600-0854.2003.00156.x.

[9] S. N. A. Puneet Khandelwal and G. Apodaca, "Cell biology and physiology of the uroepithelium", *American Journal of Physiology-Renal Physiology*, vol. 297, no. 6, F1477–F1501, 2009, PMID: 19587142. DOI: 10.1152/ajprenal.00327.2009. eprint: http://www.physiology.org/doi/pdf/10.1152/ajprenal.00327.2009. [Online]. Available: http://www.physiology.org/doi/abs/10.1152/ajprenal.00327.2009.

[10] S. A. Lewis, "Everything you wanted to know about the bladder epithelium but were afraid to ask", *American Journal of Physiology-Renal Physiology*, vol. 278, no. 6, F867–F874, 2000, PMID: 10836974. DOI: 10.1152/ajprenal.2000.278.6.F867. eprint: https://doi.org/10.1152/ajprenal.2000.278.6.F867. [Online]. Available: https://doi.org/10.1152/ajprenal.2000.278.6.F867.

[11] H. J. L. W. R. Cross I. Eardley and J. Southgate, "A biomimetic tissue from cultured normal human urothelial cells: Analysis of physiological function", *American Journal of Physiology-Renal Physiology*, vol. 289, no. 2, F459–F468, 2005, PMID: 15784840. DOI: 10.1152/ajprenal.00040.2005. eprint: http://www.physiology.org/doi/pdf/10.1152/ajprenal.00040.2005. [Online]. Available: http://www.physiology.org/doi/abs/10.1152/ajprenal.00040.2005.

[12] J. A. Glazier, A. Balter, and N. Poplawski, "Magnetization to morphogenesis: A brief history of the glazier-graner-hogeweg model", *Single-Cell-Based Models in Biology and Medicine*, p. 79, 2007. [Online]. Available: https://www.researchgate.net/profile/Ariel_Balter/publication/227073495_Magnetization_to_Morphogenesis_A_Brief_History_of_the_Glazier-Graner-Hogeweg_Model/links/00b7d52d79e94eadc7000000.pdf.

[13] F. Graner and J. A. Glazier, "Simulation of biological cell sorting using a two-dimensional extended potts model", *Phys. Rev. Lett.*, vol. 69, pp. 2013–2016, 13 Sep. 1992. DOI: 10.1103/PhysRevLett.69.2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.69.2013.

[14] J. A. Glazier and F. Graner, "Simulation of the differential adhesion driven rearrangement of biological cells", *Phys. Rev. E*, vol. 47, pp. 2128–2154, 3 Mar. 1993. DOI: 10.1103/PhysRevE.47.2128. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.47.2128.

[15] E. Stott, N. Britton, J. Glazier, and M. Zajac, "Stochastic simulation of benign avascular tumour growth using the potts model", *Mathematical and*

Computer Modelling, vol. 30, no. 5, pp. 183–198, 1999. DOI: https://doi.org/10.1016/S0895-7177(99)00156-9. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895717799001569.

[16] N. Chen, J. A. Glazier, J. A. Izaguirre, and M. S. Alber, “A parallel implementation of the cellular potts model for simulation of cell-based morphogenesis”, *Computer Physics Communications*, vol. 176, no. 11, pp. 670–681, 2007. DOI: https://doi.org/10.1016/j.cpc.2007.03.007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465507002044.

[17] T. M. Cickovski, C. Huang, R. Chaturvedi, T. Glimm, H. G. E. Hentschel, M. S. Alber, J. A. Glazier, S. A. Newman, and J. A. Izaguirre, “A framework for three-dimensional simulation of morphogenesis”, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 2, no. 4, pp. 273–288, Oct. 2005. DOI: 10.1109/TCBB.2005.46.

[18] [Online]. Available: http://www.compucell3d.org/.

[19] J. A. Izaguirre, R. Chaturvedi, C. Huang, T. Cickovski, J. Coffland, G. Thomas, G. Forgacs, M. Alber, G. Hentschel, S. A. Newman, and J. A. Glazier, “Compucell, a multi-model framework for simulation of morphogenesis”, *Bioinformatics*, vol. 20, no. 7, pp. 1129–1137, 2004. DOI: 10.1093/bioinformatics/bth050. eprint: /oup/backfile/content_public/journal/bioinformatics/20/7/10.1093 /bioinformatics/bth050/2/bth050.pdf. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/bth050.

[20] M. H. Swat, J. Belmonte, R. W. Heiland, B. L. Zaitlen, J. A. Glazier, and A. Shirinifard, *Introduction to compucell3d v3.7.4*, 2017. [Online]. Available: www.compucell3d.org/BinDoc/cc3 d_binaries/Manuals/Introduction_To_CompuCell3D_v.3.7.4.pdf.

[21] A. Torelli, P. Erben, J. Debatin, and M. Gumbel, “Proliferation and regeneration of the healthy human urothelium: A multi-scale simulation approach with 16 hypotheses of cell differentiation”.

Appendix A

Erster Anhang

Hier ein Beispiel für einen Anhang. Der Anhang kann genauso in Kapitel und Unterkapitel unterteilt werden, wie die anderen Teile der Arbeit auch.

Appendix B

Zweiter Anhang

Hier noch ein Beispiel für einen Anhang.