



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, Pauenschlag 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. 40 pages.

Mueller, Thorsten:

Einsatz eines Flux-Kompensators für Zeitreisen mit einer maximalen Höchstgeschwindigkeit von WARP 7 / Thorsten Mueller. –

Bachelor-Thesis, Mannheim: Hochschule Mannheim, 2018. 40 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 überwandten sich, umdrängten den Käfig und wollten sich gar nicht fortrühren.

Contents

| | |
|--|-----------|
| 1. Introduction | 1 |
| 1.1. Motivation | 1 |
| 1.2. Background | 2 |
| 1.2.1. Biology of the Urothelium | 2 |
| 1.2.2. CompuCell3D | 3 |
| 1.2.3. Glazier Graner Hogeweg Model | 5 |
| 1.3. Objective | 7 |
| 1.4. Outline | 8 |
| 2. State of the Art | 9 |
| 2.1. Display and Simulation of the Urothelium | 9 |
| 2.1.1. Events in the simulation | 11 |
| 2.2. Models | 12 |
| 2.3. Adhesion and cell sorting | 13 |
| 2.4. Cell Properties | 15 |
| 2.5. Fitness functions | 16 |
| 2.5.1. Arrangement fitness function | 16 |
| 2.5.2. Volume fitness function | 16 |
| 2.5.3. Overall fitness function | 17 |
| 2.5.4. Moduro Toolbox | 17 |
| 2.6. Simulation Functionalities | 18 |
| 2.6.1. Position of the stem cells | 18 |
| 2.6.2. MinMaxVolume | 18 |
| 2.6.3. Volume and TargetVolume | 19 |
| 3. Methods | 21 |
| 3.1. Lambda multipliers | 21 |
| 3.2. Abstract methods | 22 |
| 3.3. Area of stem cells on the basal membrane | 23 |
| 3.4. Target volume and target surface after mitosis | 25 |
| 3.5. Approximation Error | 26 |
| 3.6. Draw Sphere Cells | 27 |
| 3.7. Growth of a sphere cell | 31 |
| 3.8. Calculation of the volume and surface sites of a voxel sphere | 34 |

| | |
|---|-------------|
| 4. Results | 37 |
| 4.1. Draw Sphere Cells | 37 |
| 5. Conclusion | 39 |
| 5.1. Draw Sphere Cells | 39 |
| 5.2. Calculate Surface Variation Pixels to Sphere | 39 |
| 5.3. lambda values | 40 |
| 5.4. Approximation & calculation Errors | 40 |
| 5.5. CC3D | 40 |
| List of Abbreviations | vii |
| List of Tables | ix |
| List of Figures | xi |
| Listings | xiii |
| Bibliography | xv |
| Index | xix |
| A. Erster Anhang | xix |
| B. Zweiter Anhang | xxi |

Chapter 1

Introduction

1.1. Motivation

One of several requirements regarding complex life is cell adhesion. If different cells would not stick to each other the only living things would be cells. In humans, organs are made of several cells as well as epithels are made of several cells and several layers of cells. This is also the case for the urothelium, which is an epithelium, i.e. a membranous tissue which consists of one or several layers. 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 grow.

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 grow.

Bladder cancer starts to grow in the urothelium. With the grow and spread, the structure of cells sticking together is changed. In this case, the urothelium is not able to completely perform its tasks. In order to understand the urothelium, how and when bladder cancer appears observations of this epithel is necessary.

To understand the functionalities of organs and epithels, in general organisms, observations are necessary. For the urothelium this is already done, as there are a lot of different in vitro experiments about the methodology of the urothelium. After an

observation of an epithel or organ is complete, researches are able to predict how the observed organism will react in different situations. To verify these predictions a simulation is necessary. A simulation is an illustration of the reality **REF** but it can also be used to change reality in a for the research specific way to get more knowledge of the epithel, or an organism in general.

A simulation should always be as simple as possible but also not too simple **REF** Otherwise the simulation does not represent the reality. There are several programs with different algorithms for cell simulation. A popular algorithm is the Glazier-Graner-Hogeweg (GGH) model. This model is popular because it is easy to describe how cells interact with each other and it is possible to define constraints for the volume and surface of each cell.

The program CompuCell3D (CC3D) is a simulation program, which uses the GGH algorithm in its simulation. In the moduro project we use CC3D, and with the program we use the GGH algorithm.

The target of the moduro project is to predict under which circumstances bladder cancer occurs and when it is able to grow. Therefore, 16 different morphogenesis models of the urothelium were created. An overview of these models is displayed in table 2.1 at page 14. So far, all 16 models were simulated in 2D for 720 days. The results reveal that some of the models are more realistic and others are less realistic.

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

1.2. Background

1.2.1. Biology of the Urothelium

Bladder cancer is the 4th most common cancer type in men regarding to everyday-health.com, where every 36st out of 100.000 men gets it [2]. Bladder cancer usually starts with some cells in the bladder, which grow uncontrolled. From these cells, the tumor can spread further into surrounding areas [3]. The most common bladder cancer type is the urothelial carcinoma [3].

The bladder is located in the lower urinary tract and consists 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 enlarge and downsize its size. This is done by the largest cells of the urothelium, the umbrella cells. Since the umbrella cells are in direct contact with the bladder, it is their task to change size and form during the grow and shrink process of the bladder. Birder described the urothelium as “... a responsive structure capable of detecting physiological and chemical stimuli and releasing a number of signaling molecules.” [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].

To receive a better overview the different cell types are explained in the following paragraph. In figure 1.1 at page 4 a simplified illustration of the urothelium with its different cell types and cell layers is provided.

The umbrella cells, also called superficial cells, are connected directly with the bladder and have an average diameter of 25 up to 250 μm [5], [9].

Below these cells the intermediate cells are located. With an average diameter of 10 up to 20 μm [5], [9], they are smaller than the umbrella cells. There are at least three and up to five layers of the intermediate cells **REF**

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 μm [4], [9].

The urothelium consists of 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 there is one layer of umbrella cells.

1.2.2. CompuCell3D

CC3D is an open-source program, which provides a simulation environment for multi- or single-cell-based modeling of tissues, organs and organisms [12]. To do

1. Introduction

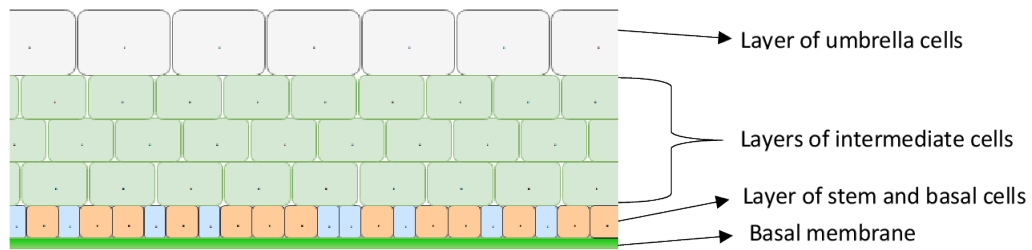


Figure 1.1.: A simplified illustration of the urothelium. At the very bottom there is the basal membrane. Above the membrane the stem and basal cells are displayed. The blue cells represent the stem cells, the red cells display the basal cells. Above the layer out of these two cell types, the urothelium contains several layers of intermediate cells. At the top of the urothelium there is an layer of umbrella cells.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 3 | 3 | 5 | 5 | 5 |
| 1 | 1 | 1 | 3 | 3 | 3 | 5 | 5 |
| 2 | 2 | 1 | 3 | 4 | 5 | 5 | 5 |
| 2 | 2 | 2 | 4 | 4 | 4 | 7 | 7 |
| 6 | 2 | 4 | 4 | 4 | 4 | 4 | 7 |
| 6 | 6 | 4 | 4 | 4 | 7 | 7 | 7 |
| 6 | 6 | 9 | 9 | 9 | 9 | 8 | 7 |
| 6 | 6 | 9 | 9 | 9 | 8 | 8 | 8 |

Figure 1.2.: A square lattice in 2D. The same digits represent one cell ($\sigma(\vec{i})$), whereas the different colors represent different cell types $\tau(\sigma)$.

so, CC3D uses the GGH model in its simulation. CC3D provides the possibility to create programs for the simulation, e.g. cell growth, mitosis, apoptosis or necrosis scripts, in python, C++ or in CC3DML, which is their own Markup Language. With such programs CC3D allows the user to modify the behavior of the simulation for a specific purpose. CC3D uses the GGH approach, explained in section 1.2.3 at page 5. It allows the user to choose between two cell-lattice types, i.e. a presentation of the pixels or voxels of a cell at a specific position in the simulation field. By default it uses a square-lattice of single pixels for each dimension, an example therefore is displayed in figure 1.2 at page 4. 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 [13], 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:

$$\mathcal{H}_{boundary} = \sum_{\vec{i}, \vec{j}} J(\tau(\sigma(\vec{i})), (\tau(\sigma(\vec{j}))) (1 - \delta(\sigma(\vec{i}), (\sigma(\vec{j}))) \quad (1.1)$$

This equation is a part of the equation of the GGH model. The formula of the GGH model, and with it this equation, is explained at section 1.2.3 at page 5.

There are two ways to extend this form, it is possible to add a volume or a surface constraint for each cell. During each MCS an index-copy attempt takes place [13]. Therefore, 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 succeeds and takes place if this index copy attempt decreases the effective energy [13]. Each MCS the program tries to minimize the effective energy with index copy attempts.

1.2.3. Glazier Graner Hogeweg Model

Since there are several formulas and models developed from Glazier and Graner, this subsection briefly describes these models and explains the Cellular Potts Model (CPM) and GGH model.

The GGH model is widely used in biological simulations, since it provides a good flexibility, extensibility and it is easy to use [14]. Glazier and Graner developed a 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 CPM [14]–[16]. Glazier and Graner extended the CPM in a way that also volume constraints are considered for the hamiltonian, see following form:

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

The hamiltonian of equation 1.2 describes the effective energy for the extension of the CPM model. The first sum describes J of all cells, the adhesion energy between different cells. Therefor, every cell has a specific cell type $\tau(\sigma)$ [15], [16]. Each cell is placed onto a lattice with a spin $(\sigma(\vec{i}))$ for every given dimension [14], [15]. The adhesion energy between cells is only considered if the kroenecker delta is 0. Thus, the surface energy between cells is considered if $\delta(\sigma, \sigma') = 0$ [14]–[19].

With the second sum over all cells the volume of each cell is now considered within the effective energy. The user is now able to set a target volume $V_{target}(\tau(\sigma))$ for each cell, which the cell should have, and a multiplier λ_{vol} for the deviation between the current volume $(\tau)v(\sigma)$ and the target volume. During the simulation this deviation is tried to be kept as small as possible for every cell in order to keep the effective energy as small as possible.

Together with Hogeweg they further developed the created extension of the CPM. The further developed model is called GGH model. The main extension is that the user is now able to add surface area constraints [14]–[16] as well as to use a negative boundary energy [14]. With the surface constraint the equation for the effective energy of the GGH model is:

$$\begin{aligned}\mathcal{H}_{GGH} = & \sum_{\vec{i}, \vec{j}} J(\tau(\sigma(\vec{i})), (\tau(\sigma(\vec{j}))) (1 - \delta(\sigma(\vec{i}), (\sigma(\vec{j}))) \\ & + \sum_{\sigma} \lambda_{vol} (\tau)v(\sigma) - V_{target}(\tau(\sigma)))^2 \\ & + \sum_{\sigma} \lambda_{sur} (\tau)s(\sigma) - S_{target}(\tau(\sigma)))^2\end{aligned}\quad (1.3)$$

In addition to the hamiltonian of equation 1.2 is the surface constraint. It has the same principle as the volume constraint. Thus, the user is able to define a target surface $S_{target}(\tau(\sigma))$ for each cell and a multiplier λ_{sur} for the deviation between

the target and the actual surface $(\tau)s(\sigma)$ of each cell. Since the volume and surface constraint are included in the effective energy, it should be possible to use these two parts of the effective energy to shape the cells.

Beside the surface constraint the new model 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 [14].

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 lattice sites sharing the same cell index [...], a set of *internal cell states* for each cell [...], and a set of *auxiliary fields* ..." [14].

The GGH model has the advantage that "Initial conditions emulating a particular biological configuration rather than random initial conditions." [14] and it has now biologically motivated properties instead of physically motivated properties [14].

1.3. Objective

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 have to be invented and developed new whereas for other functionalities it is enough to modify these.

The result of this bachelor thesis will be presented with an realistic model of the 2D simulations. This model is chosen 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 thesis, this work 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 consists of several models and a program, both are written in python. The models include properties of the specific model, e.g. adhesion energy between cells or the possibilities of the new cell types after mitosis. The application modifies the cell behavior, e.g. it checks when a mitosis takes place, how fast the cell will grow, etc.. The program also calculates if the current model is a realistic one or not, after the calculation these data will be written in to an file which can be read out by the 'Moduro-Toolbox'.

2.1. Display and Simulation of the Urothelium

In CC3D we simulate an urothel of the size of $333\text{ }\mu\text{m}$ for the x-axis and $100\text{ }\mu\text{m}$ for the y-axis. Because, CC3D has its constraints about the size if we use one core of the processor the size of the simulation will be rescaled to its maximum limitations of $267\text{ }\mu\text{m}$ for the x-axis and $80\text{ }\mu\text{m}$ 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 [13]. I.e. a cell is in both grids, therefore 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 both will apply the new volume of their calculation but the user do not know which

2. State of the Art

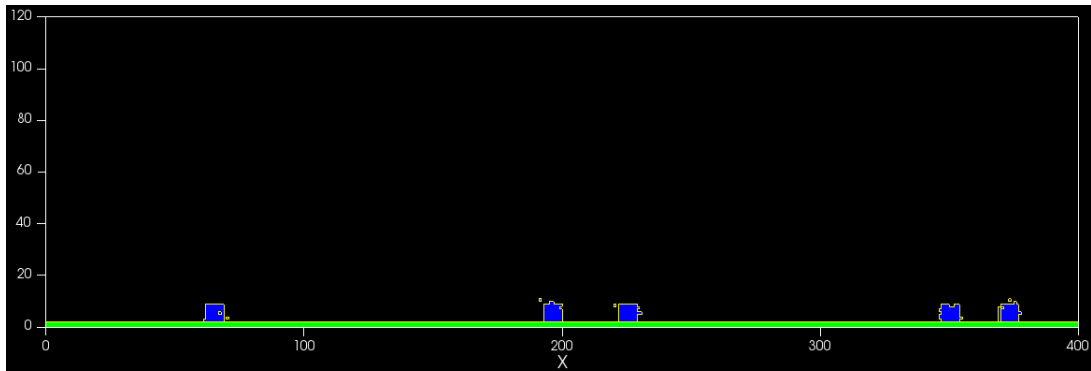


Figure 2.1.: Initial state of an 2D simulation of the model SpaPcdbCdiIn

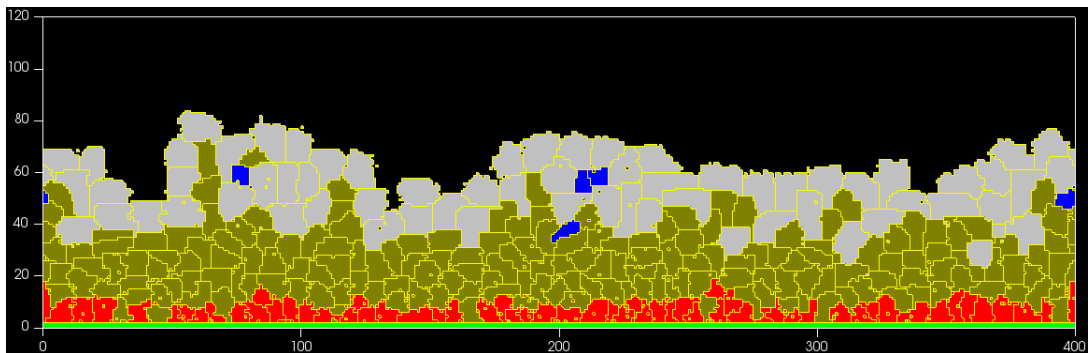


Figure 2.2.: 2D Simulation of the model SpaPcdbCdiIn after 33 days

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 10. 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 10.

In order to provide a realistic simulation, the simulation has to be not too easy and not too 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 multiplier 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 disappear.

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 probability to become evil, in the current simulation it is not considered that cells mutate since the probability 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 disappears.

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 [20]. In the program a flag for necrosis is set and the cells will disappear 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" [20] 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" [20]. 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 [20] and are displayed in table 2.1 at page 14.

2.3. Adhesion and cell sorting

During morphogenesis 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 [20] and is displayed in table 2.2 at page 15. 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.

2. State of the Art

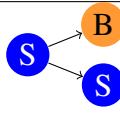
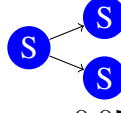
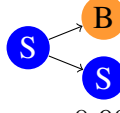
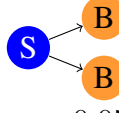
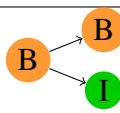
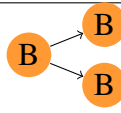
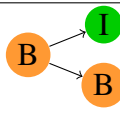
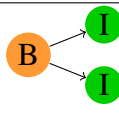
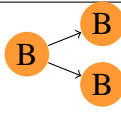
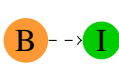
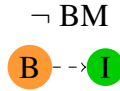
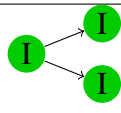
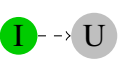
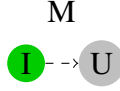
| Type | ID | Description | Model |
|--------------------|------|---|---|
| Stem cells | SSD | Stem cell-like division |  |
| | SPA | Stem cell population asymmetry |    $p_s = 0.05$ $p_a = 0.90$ $p_s = 0.05$ |
| Basal cells | BSD | Stem cell-like division in basal cell |  |
| | BPA | Basal cell population asymmetry |    $p_s = 0.05$ $p_a = 0.90$ $p_s = 0.05$ |
| | BPCD | Proliferation and contact differentiation of basal cells |  and $\neg \text{BM}$  |
| | BCD | Only contact differentiation of basal cells | $\neg \text{BM}$  |
| Intermediate cells | IPCD | Proliferation and contact differentiation of intermediate cells |  and M  |
| | ICD | Only contact differentiation of intermediate cells | M  |

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 15 [20]. 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 [20]. 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 [20], 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**

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

In this equation L_B and L_U are boolean values, i.e. they have the value 0 or 1 [20]. 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 [20]. lib is the amount of layers in betwenn the first and the last layers [20]. L_I contains the amount of layers, which consits only intermediate cells [20]. Therefore, $lib - L_I$ describes the amount cells which are not in there intended layer [20].

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 [20].

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 μm [20]. Therefore the formula is:

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

V_{Si} and V_{Ii} describes the *should* and the actual *is* volume of a specific cell type i [20].

2.5.3. Overall fitness function

The overall fitness function calculates the total fitness out of the volume and the arrangement fitness function. Therefore the average of both functions is calculated [20]. The function is the following:

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

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 [20]. Therefore, the formula is:

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

$e + 1$ indicating the amount of calculations of the overall fitness function [20].

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. Therefore 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 necessary. This is done by the 'MinMaxVolume' in the cell Dictionary. This 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 [20]. 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

Methods

This section has the purpose to give an overview over all changes in the project which occurred during this bachelor thesis. These changes include some small improvements of the application as well as how to draw a sphere cell. First minor changes are briefly described, deeper in this section the more effortful changes are explained.

3.1. Lambda multipliers

In the project there were several places, where the multipliers λ_{vol} and λ_{sur} are calculated, but they are only set when a cell is initialized. The multiplier for the volume is set a second time in the necrosis event, if the program models that a cell dies.

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 program and they do not influence or set the λ_{vol} or λ_{sur} which CC3D uses, these two values are deleted. Moreover, in one place there were methods to calculate the lambda values. Since these methods are not used in the project and they had a multiplier itself to calculate the multiplier for the specific part of the effective energy, they are deleted as well.

In the project the multipliers λ_{vol} and λ_{sur} are now set each time when a cell is initialized. A second time the multiplier λ_{vol} is set, is if necrosis takes place. Because now the unnecessary methods and values for the multipliers values are removed, no confusion about which constraint values in the program are used will come up.

3.2. Abstract methods

In the program, in several classes there were methods, which are not used. These methods are not used in the class, in which they are written, due to polymorphism. They are used in classes, which inherit from the class where they are written. Polymorphism is a method out of Object Oriented Programming (OOP). Another technique out of OOP is the use of abstract classes and abstract methods. To explain polymorphism, abstract classes or abstract methods in detail, take too much space out of this bachelor thesis. Thus, the change in the program is explained in the following.

In the project there are several classes which inherit from each other. In these classes there were the same methods implemented. Since, it is not required that all classes have to have the same methods, these are redundant methods, as long as they do not differ in their functionality. For such methods it is possible to declare as an abstract method. An abstract method contains only the method construct, but no functionality. Due to polymorphism the class in which such an abstract method is implemented is used to call this method.

In python there is a library 'abc'. With this library it is possible to declare an abstract method. To do so every class with one or more abstract methods need to initialize a special class variable. With this variable python is able to recognize that there are abstract methods included.

```
class ModelConfig(object):  
    __metaclass__ = ABCMeta
```

Listing 3.1: The initialization of a class variable which is required by python in order to detect abstract methods and abstract classes.

The listing above displays how the class variable has to be initialized in order that abstract methods are recognized. If a class has at least one abstract method, it is an abstract class. In python abstract classes can include implemented methods as well as abstract methods.

```
@abstractmethod  
def _createExecConfig(self):  
    pass
```

Listing 3.2: Declaration of an abstract method.

The listing above is an example of an abstract method in python. Using abstract methods creates more structure and clarity within the project. Therefore, the ab-

struct declarations in this project created a much better clarity of the structure of the project and of the methods, which were used in the classes.

3.3. 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 [20]. In the project the calculation of the amount of stem cells for two dimensions were correct but without an 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. Therefore, in two dimensions the area of the stem cells should be calculated by using the cell diameter. An illustration therefor is displayed in figure 3.1. For three dimensions it is possible to calculate the area of a circle with the formula $A = \pi \cdot r^2$, because a sphere in 2D is a circle, 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 3.2.

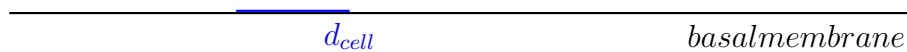


Figure 3.1.: Considered area to spread the stem cells in 2D with an example of one stem cell placed on the basal membrane. Because only the x-axis is displayed we need to calculate the diameter of a cell.

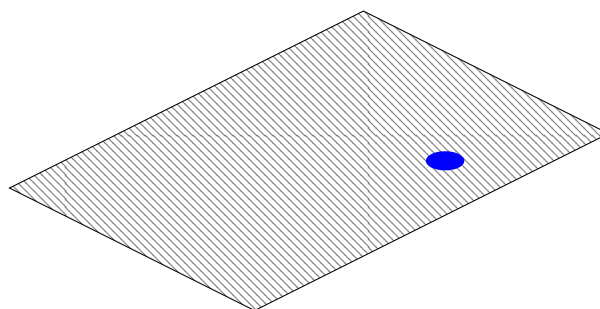


Figure 3.2.: Considered area to spread the stem cells in 3D with an example of one stem cell placed on the basal membrane. The hatched area displays the basal membrane and the circle represents one stem cell. This stem cell is a circle, because in the calculation of the amount of stem cells the y-axis is negligible.

3. Methods

Table 3.1.: Possible approximation error by not rounding the result of equation 3.2. The first column describes the length of the basal membrane, the values are in μm . In the second column the result of equation 3.2 is displayed. In the third column the result of the second column is rounded up in the first row. In the second row this result is casted. The fourth column displays how much space, in μm , the rounded or casted result of equation 3.2 require. The last column displays the physical space required in percentage to the basal membrane.

| xLength of the basal membrane | Result of equation 3.2 | Amount of stem cells | Area used of stem cells in μm | relative used area |
|-------------------------------|------------------------|----------------------|--|--------------------|
| 200 | ~ 2.66 | 3 | 27 | 13.5% |
| 200 | ~ 2.66 | 2 | 18 | 9% |

With the following two equations it is possible to calculate the amount of stem cells in two dimensions. $A_{stemcells}$ refers to the area which can be used for stem cells. Therefore, it is 12% of the basal membrane. c is a constant, which describes 12% of an object. Thus, $c = 0.12$.

$$A_{stemcells} = xLength \cdot c \quad (3.1)$$

$$Amount_{StemCells} = \frac{A_{stemcells}}{d_{cell}} \quad (3.2)$$

Equation 3.1 ensures that only 12% of the given area is used. Formula 3.2 then calculates the amount of stem cells on this given area. Because 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 casted, i.e. the decimal digits are cut off.

As table 3.2 displays, it is important to round the result. Otherwise there would be an approximation error and as a result a calculation error.

For three dimensions the equation 3.1 has to be extended, because the z-axis is now also considered in the calculation of the amount of stem cells on the basal membrane. An illustration therefor is figure 3.2. Equation 3.2 has to be modified, because now the area of an circle, instead of the diameter, is used for the calculation. Therefore, the equations to calculate the amount of stem in three dimensions are:

$$A_{stemcells} = (xLength \cdot zLength) \cdot c \quad (3.3)$$

$$Amount_{StemCells} = \frac{A_{stemcells}}{\pi \cdot r^2} \quad (3.4)$$

The result of formula 3.4 has to be rounded as well, otherwise the program would include rounding errors. These calculations are now included in the program.

3.4. Target volume and target surface after mitosis

Mitosis of the cell is simulated by CC3D. In the simulation mitosis is simulated as the following: one cell splits into two cells. The cell which splits dies and then two new cells out of the died cell are created.

In the program we specify and check if and when a cell splits. CC3D decides where the cell splits and calculates the volume and 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 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 might not split in the middle. In the case of mitosis, it may be a source of error to set the target volume, of the two created cells, without the knowledge of the volume. Because the target volume of the two new cells are set without considering the current volume of the cell, it occurred that one of the new cells, after mitosis, had a target volume which was smaller than the current volume.

After mitosis both cells are initialized. Thus, the target volume and target surface is calculated and set. After the initialization of the cells is done, the target volume of the new cells is set to the target volume divided by 2 of the cell before mitosis.

Because the target volume is calculated out of the current volume of the cell during the initialization process, the calculation of the target volume is removed out of the mitosis event. Now the target volume and surface after mitosis is set only during the initialization process of the new cells.

With this change of the program the case that some cells have a smaller target volume than the current volume disappeared. In the program now, the target volume and target surface after mitosis is calculated and set dependent on the of CC3D given volume.

3.5. Approximation Error

Because the project includes conversions from μm into voxels and the amount of pixels, e.g. for the surface of a cell, has to be set as data type 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 unit μm are saved with the data type float, i.e. a number with decimal digits. To set these values as a whole number the values are casted into the data type integer.

In the project whenever a conversion from μm into voxels is done 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 removed, because the cast is the very last step. It is possible that in the program still have some rounding errors, but these can not be removed because CC3D requires the amount of voxels as a whole number and the calculation is with decimal digits.

In the function displayed below, the cast and the rounding of the result is done in the last step of the calculation.

```
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 3.3: Function to calculate the volume of a sphere in voxels out of a given physical volume. First out of the given physical volume the radius is calculated. Then the radius is converted into the voxel unit. Next the volume of the voxel sphere is calculated and as last step the result is rounded and casted.

One use of the displayed function is to calculate the voxel volume out of the physical volume of a cell. In the simulation a minimum and a maximum volume for each cell type is calculated. These values are used in the calculation to determine if mitosis takes place or not.

For the basal cell the minimum volume is $381\mu\text{m}$ and the maximal volume is $523\mu\text{m}$. In table 2.3 at page 15 the constraints of the different cell types in μm

are displayed. In the table 3.2 three different possible calculations of the conversion of a volume in μm into a volume in voxels are presented.

Table 3.2.: Three different ways to calculate the voxel volume out of a given physical volume. The first column describes the physical volume in μm . In the second column the radius in μm out of the volume is calculated. Next, the radius will be used as it is, casted or rounded up. In the fourth column the exact result of the volume in voxel is presented and in the last column the rounded result of the voxel volume is displayed.

| Volume μm | in | radius in μm | radius used in further calcu- lation | not rounded result in vx | rounded result in vx |
|-------------------------|----|-------------------------|--|-----------------------------|-------------------------|
| 381 | | 4.49 | 4.497 | 1285.67 | 1286 |
| 381 | | 4.49 | 4 | 904.77 | 905 |
| 381 | | 4.49 | 5 | 1767.15 | 1767 |

In the table, the first row calculates the voxel volume as it is done in project right now. Thus, rounding and casting is the last step in the calculation. In the second row the radius is calculated and casted. Then this casted radius is used for the further calculation. The third row calculates the radius and rounds it immediately. This rounded radius is then used for the further calculation.

As the table displays, there is a significant difference in all three results. Because such a calculation is used to determine when a cell splits as well as it calculates the growth per MCS, it influences the result of a simulation. Thus, it is important to round and cast the result as very last step in the calculation.

3.6. Draw Sphere Cells

Since a sphere as a cell is an approximation to a cell in the urothelium, it is a possible technique to draw and further simulate a cell as a sphere.

To be able to draw a sphere out of voxels it is required that a cuboid lays around the sphere, as it is displayed in 3.3. The cuboid has to be at least as large as $2 \cdot \text{radius}$ of the sphere. In addition, the cuboid should not be larger than necessary, otherwise there would be unnecessary computable cost. The illustration in figure 3.4 at page 30 display the perfect size of a square and a circle. These 2D objects are chosen to display the boundaries of a circle in a square as well as the boundaries would be in three dimensions.

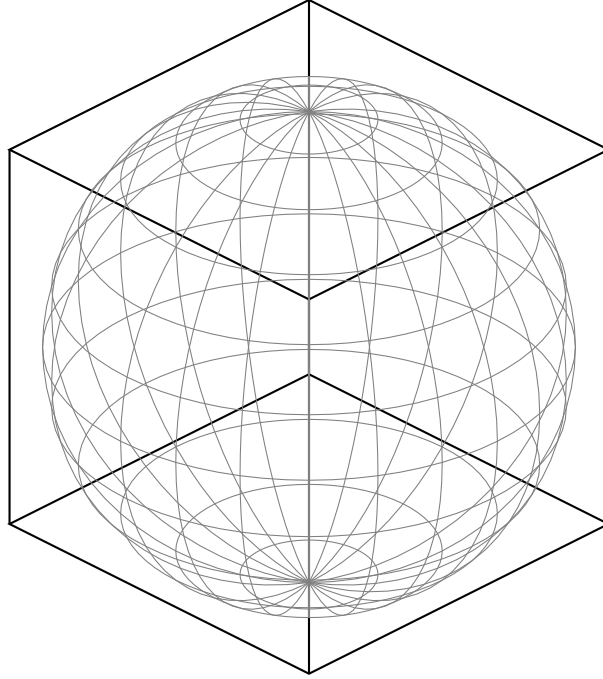


Figure 3.3.: A cuboid layed around a sphere

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.

Because we are using the square lattice, 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 equation 3.5 provides a mechanism in which every point within the sphere can be calculated.

$$\sqrt{(x_r - x_0)^2 + (y_r - y_0)^2 + (z_r - z_0)^2} \leq radius \quad (3.5)$$

In the equation above 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 coordinate is smaller or equal to the radius the current point is within the sphere otherwise it would be outside of the sphere. Because a voxel itself contains several points, this equation, in this case, 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 accuracy of the equation above.

In the program it is possible to calculate the corner length of a voxel. Thus, it is possible to decide if the center of a voxel is at the inside or outside of a sphere. How the center of a voxel is calculated is explained in the following.

Since, the radius of the sphere is known, whether it is known or it is calculated out of the volume or the area of a sphere, the diameter of the cuboid can be calculated. Since the voxel density is known in the simulation, the corner length of each voxel can be calculated as it is displayed in equation 3.6.

$$c = \frac{2 \cdot radius}{voxeldensity} \quad (3.6)$$

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 axes. The following three formulas in equation 3.7 display such calculations.

$$\begin{aligned} x_c &= x_r + \frac{x_{r+c} - x_r}{2} \\ y_c &= y_r + \frac{y_{r+c} - y_r}{2} \\ z_c &= z_r + \frac{z_{r+c} - z_r}{2} \end{aligned} \quad (3.7)$$

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 3.7 it is now possible to consider the center of a voxel in the decision if the voxel is in- or outside the sphere, as it is displayed in equation 3.8.

$$\sqrt{(x_c - x_0)^2 + (y_c - y_0)^2 + (z_c - z_0)^2} \leq radius \quad (3.8)$$

To draw a cell as a sphere a new function had to be created. This function is named `addSphereCell` as it is displayed below. In the method all required points, the start- and end points as well as the center, of all three axis are given in μm . Thus, these points as well as the radius, which is also given to the function, and the steplength, i.e. the corner length of a voxel, are converted into the voxel unit.

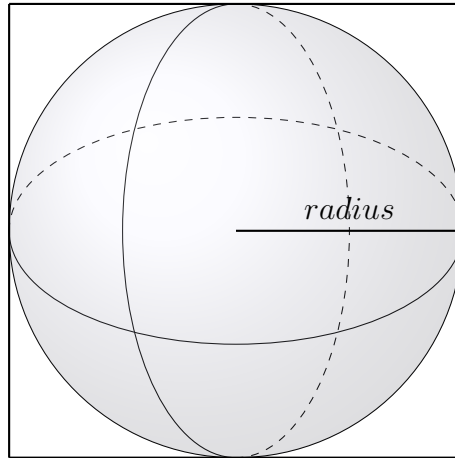


Figure 3.4.: A cuboid with minimal size layed around a sphere

With all necessary points in the voxel unit it is now possible to iterate over all three axis and check for every voxel, if it is included in the sphere or not.

```
def _addSphereCell(self, typename, xPos, yPos, zPos, radius, steppable):

    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 = self.execConfig.calcFloatPixel(1)

    # 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: Function to draw a cell as a sphere. First all required points for the calculation are converted into the voxel unit. Then over each axis of the cuboid it is iterated. During these iterations for each voxel the distance to the center of the cuboid and sphere is calculated and then it is checked if the voxel is within the sphere or not. If the voxel is a part of the sphere it will be added to the sphere.

With this created function it is now possible to draw a cell as a sphere, as it is displayed at figure xy.

To draw the cell as a sphere is the first step to have sphere cells in the simulation. Two major 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.

3.7. Growth of a sphere cell

Since in the project morphogenesis of the urothelium is simulated, the drawn sphere cells are required to growth.

In the simulation the current volume and surface of a cell is calculated by CC3D. Every user is able to influence these values by setting the target volume and target surface and the proper multiplier, λ_{vol} and λ_{sur} , for the specific part of the effective energy. Because the goal of the simulation is to minimize the effective energy, the cell will change the current volume and surface in the direction of the set target volume and target surface.

To model the growth of a cell, the growth per day of the volume for the different cell types is set. Each MCS the growth of one MCS is calculated and applied, 500 calculations steps are one day. After the growth of the volume is calculated and set as new target volume of the specific cell, a new target surface depending on the target volume is calculated and set.

In the program the target surface is calculated out of a real sphere and then multiplied by a factor. Since the surface of a sphere of voxels is larger than the surface of a real sphere, it is necessary to find the correct factor, by which the surface of a sphere has to be multiplied, in order to calculate the surface of a sphere of voxels.

For simplicity reasons, a first approximation of the factor is calculated in 2D. Thus, a circle and a square filled with pixels are used. An example therefor is displayed in figure 3.5.

Figure 3.6 displays a possible approximation, with which it is possible to calculate the factor for the surface.

Consider in figure 3.6 the left square as a pixel, it could be any pixel of the figure 3.5 in which the circle goes through. Thus, the sides of the square are the same. The blue line represents the circle, in a way it could go through the pixel. An

3. Methods

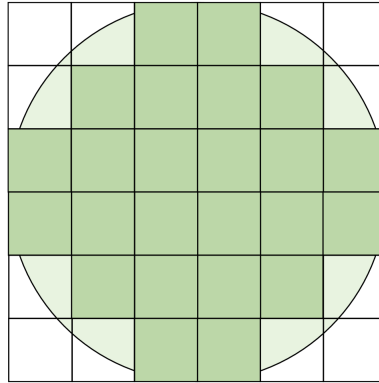


Figure 3.5.: A circle in a possible pixel presentation. All small squares present pixels. The colored squares present the pixels, which are included in the pixel presentation of the circle.

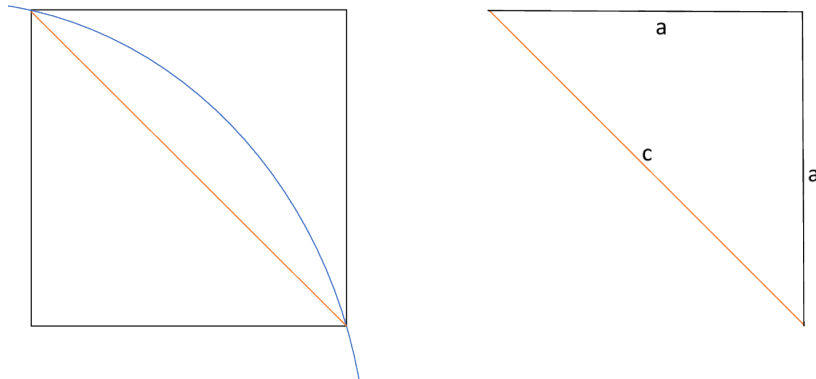


Figure 3.6.: The left part of the illustration displays a pixel at the surface, in which the circle goes through. The blue line represents the circle. As an approximation to the surface line a diagonal line in the square is drawn. With this diagonal line a isosceles, right angular triangle appears and it is possible to calculate the additional surface of one pixel. This isosceles, right angular triangle is displayed at the right side of this figure.

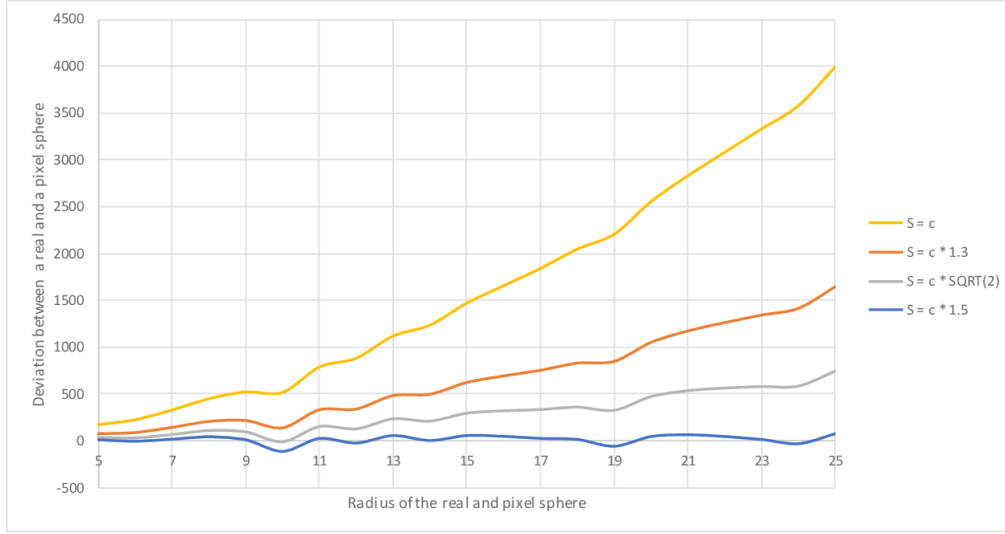


Figure 3.7.: Deviation between the surface of sphere of voxels and a real sphere. Each deviation is calculated with a different factor of the surface of the real sphere. S refers to the surface of the sphere of voxels and c refers to the surface of a real sphere. Thus, $c = 4 \cdot \pi \cdot r^2$. At the x-axis the radius of the sphere and the sphere of voxels is displayed. At the y-axis the deviation between both surfaces is represented.

approximation is to insert a diagonal line. With this diagonal line a isosceles, right angular triangle is created as it is displayed at the right side of figure 3.6.

The following formula provides a way to calculate the sides of the isosceles, right angular triangle.

$$c^2 = 2 \cdot a^2 \quad (3.9)$$

This formula can be used to calculate the side a , as it is displayed in the following equation.

$$a = \sqrt{\frac{c^2}{2}} \quad (3.10)$$

$$a = \frac{c \cdot \sqrt{2}}{2}$$

Since the surface of a pixel has two corner sides, the equation to calculate the surface of a pixel sphere is

$$S = 2a \quad (3.11)$$

$$S = c \cdot \sqrt{2}$$

This formula applies to all pixels, which are at the surface of the pixel sphere, c is considered as the surface of the circle. Thus $c = \pi \cdot r^2$.

The calculated factor $\sqrt{2}$ was tested for a sphere surface. Because, this is an approximation to the surface of a pixel sphere, the tenth part beside this factor are also tested mathematically. As figure 3.7 displays the approximation with factor $\sqrt{2}$ still has some deviation. Almost no deviation between the surface of a sphere and a surface of a sphere of voxels is with a factor of 1.5. Thus, this factor is applied for the calculation of the target surface.

3.8. Calculation of the volume and surface sites of a voxel sphere

In order to calculate the voxel volume and the surface sites of a voxels sphere on our own, I created an algorithm for this problem. Otherwise for every measurement of the volume and surface of the sphere of voxels a new simulation in CC3D had to be started.

The algorithm requires the radius of the sphere, this radius has to be a whole number. For the tenths part between two whole numbers the result of the algorithm and CC3D deviate. Since a sphere and a cuboid are both symmetrical, it is possible to split both into 8 pieces. Doing so only one eighth of the of the cuboid has to be calculated in order to determine the volume and the surface of the sphere.

To calculate which voxels are in the voxel sphere, the algorithm creates a x, z, y matrix. In the following calculations and decisions the center of each voxel is used. For every x,z voxel in the eighth of the cuboid the y constraint of the circle is calculated. Next, every voxel at the x,z coordinate is checked if the center of the voxel is within thy constraint or not. Thus, every voxel at a x,z coordinate is checked if its center is within the sphere or not. This is repeated until every x,z coordinate of the eighth of the cuboid is calculated. Every time a voxel is within the sphere, it will be marked in the matrix.

To receive the volume of the voxel sphere all marked entries in the created matrix are count. Then, this result is mirrored for all three axis. Thus, the amount of marked voxels in the matrix is multiplied by 8, which equals to one multiply by 2 for each axis. Therefore $V = ((Voxels \cdot 2) \cdot 2) \cdot 2$. To receive the surface sites of the voxels at the surface, every marked entry is checked if the voxel at x+1, z+1 or the y+1 is either out of bounce, i.e. if it is outside the cuboid, or outside the sphere. Every of the three conditions is checked for every voxel within the surface. If one condition is true, the amount of surface sites is increased by 1. In the end the result,

like the volume, has to multiplied by 8, in order to mirror the result for all three axis.

Chapter 4

Results

ergebnisse (screenshots source code)

Abbildungen mit den Abweichungen von Volumen Surface von PixelKugel zu einer echten Kugel

Kugel füllt zuerst die ecken aus -> dann wächst der würfel nach außen

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 **exponentially** 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 $1\text{ }\mu\text{m}$. As an example, if the voxel density = 1 than it represents $1\text{ }\mu\text{m}$, if the voxel density = 2 than one voxel represents $0.5\text{ }\mu\text{m}$ and so on. In a simulation with $100\text{ }\mu\text{m}$ at the x-axis, $80\text{ }\mu\text{m}$ at the y-axis and $50\text{ }\mu\text{m}$ 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

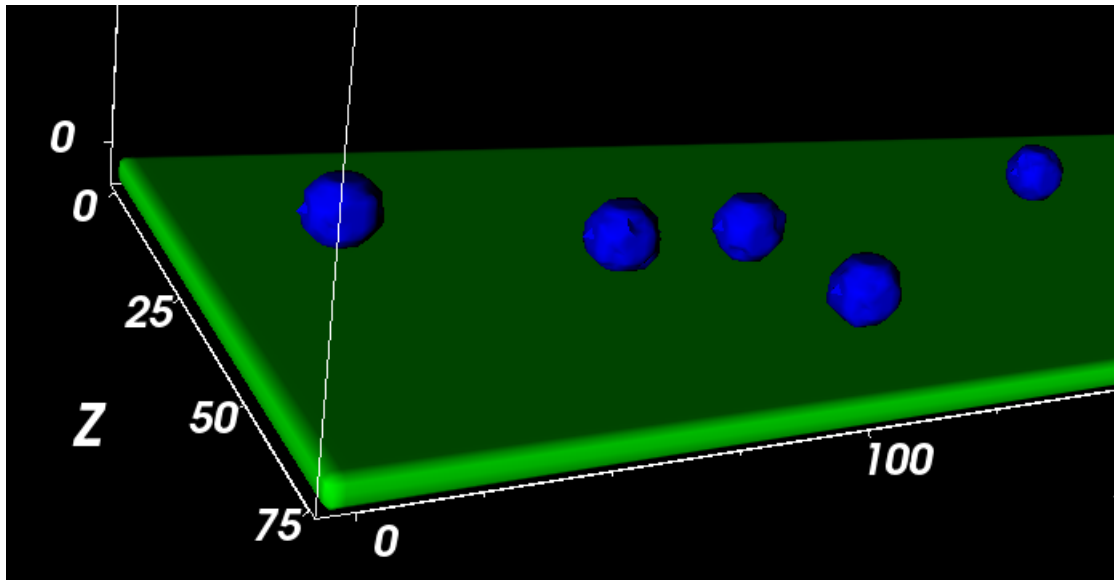


Figure 4.1.: A cell drawn as a sphere

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 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 accuracy 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**
- $400\text{vx} \cdot 400\text{vx} \cdot 400\text{vx} = 107171875\text{vx}$ 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 and CC3D crashes
- no Debug -> testing with print commands at command line

List of Abbreviations

GGH Glazier-Graner-Hogeweg

CC3D CompuCell3D

MCS Monte Carlo Step

CPM Cellular Potts Model

EPM Extended Potts Model

OOP Object Oriented Programming

List of Tables

| | | |
|------|--|----|
| 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. | 14 |
| 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. | 15 |
| 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. | 15 |
| 3.1. | Possible approximation error by not rounding the result of equation 3.2. The first column describes the length of the basal membrane, the values are in μm . In the second column the result of equation 3.2 is displayed. In the third column the result of the second column is rounded up in the first row. In the second row this result is casted. The fourth column displays how much space, in μm , the rounded or casted result of equation 3.2 require. The last column displays the physical space required in percentage to the basal membrane. | 24 |
| 3.2. | Three different ways to calculate the voxel volume out of a given physical volume. The first column describes the physical volume in μm . In the second column the radius in μm out of the volume is calculated. Next, the radius will used as it is, casted or rounded up. In the fourth column the exact result of the volume in voxel is presented and in the last column the rounded result of the voxel volume is displayed. | 27 |

List of Figures

| | | |
|------|--|----|
| 1.1. | A simplified illustration of the urothelium. At the very bottom there is the basal membrane. Above the membrane the stem and basal cells are displayed. The blue cells represent the stem cells, the red cells display the basal cells. Above the layer out of these two cell types, the urothelium contains several layers of intermediate cells. At the top of the urothelium there is an layer of umbrella cells. . . . | 4 |
| 1.2. | A square lattice in 2D. The same digits represent one cell ($\sigma(\vec{i})$), whereas the different colors represent different cell types $\tau(\sigma)$ | 4 |
| 2.1. | Initial state of an 2D simulation of the model SpaPcdbCdiIn | 10 |
| 2.2. | 2D Simulation of the model SpaPcdbCdiIn after 33 days | 10 |
| 3.1. | Considered area to spread the stem cells in 2D 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. . . . | 23 |
| 3.2. | Considered area to spread the stem cells in 3D with an example of one stem cell placed on the basal membrane. The hatched area displays the basal membrane and the circle represents one stem cell. This stem cell is a circle, because in the calculation of the amount of stem cells the y-axis is negligible. | 23 |
| 3.3. | A cuboid layed around a sphere | 28 |
| 3.4. | A cuboid with minimal size layed around a sphere | 30 |
| 3.5. | A circle in a possible pixel presentation. All small squares present pixels. The colored squares present the pixels, which are included in the pixel presentation of the circle. | 32 |
| 3.6. | The left part of the illustration displays a pixel at the surface, in which the circle goes through. The blue line represents the circle. As an approximation to the surface line a diagonal line in the square is drawn. With this diagonal line a isosceles, right ancular triangle appears and it is possible to calculate the additional surface of one pixel. This isosceles, right ancular triangle is displayed at the right side of this figure. | 32 |

| | | |
|------|--|----|
| 3.7. | Deviation between the surface of sphere of voxels and a real sphere. Each deviation is calculated with a different factor of the surface of the real sphere. S refers to the surface of the sphere of voxels and c refers to the surface of a real sphere. Thus, $c = 4 \cdot \pi \cdot r^2$. At the x-axis the radius of the sphere and the sphere of voxels is displayed. At the y-axis the deviation between both surfaces is rep- resented. | 33 |
| 4.1. | A cell drawn as a sphere | 38 |

Listings

| | | |
|------|--|----|
| 2.1. | stem cell position | 18 |
| 2.2. | MinMaxVolume | 18 |
| 2.3. | set target Volume and Surface of a cell | 19 |
| 3.1. | The initialization of a class variable which is required by python in order to detect abstract methods and abstract classes. | 22 |
| 3.2. | Declaration of an abstract method. | 22 |
| 3.3. | Function to calculate the volume of a sphere in voxels out of a given physical volume. First out of the given physical volume the radius is calculated. Then the radius is converted into the voxel unit. Next the volume of the voxel sphere is calculated and as last step the result is rounded and casted. | 26 |
| 3.4. | Function to draw a cell as a sphere. First all required points for the calculation are converted into the voxel unit. Then over each axis of the cuboid it is iterated. During these iterations for each voxel the distance to the center of the cuboid and sphere is calculated and then it is checked if the voxel is within the sphere or not. If the voxel is a part of the sphere it will be added to the sphere. | 30 |

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] [Online]. Available: <http://www.compucell3d.org/>.
- [13] 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/cc3d_binaries/Manuals/Introduction_To_CompuCell3D_v.3.7.4.pdf.
- [14] 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.
- [15] 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>.

-
- [16] 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>.
- [17] 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](https://doi.org/10.1016/S0895-7177(99)00156-9). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0895717799001569>.
- [18] 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>.
- [19] 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.
- [20] 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.