Internal Report 2014-number July 2014

Universiteit Leiden

Opleiding Informatica

Dynamic 3D visualization of Mycobacterium marinum infection modelling in zebrafish using the Petri Net formalism.

Jeroen van den Heuvel

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

Calvalho et al modelled the infection of *Mycobacterium marinum* in zebrafish in the Petri Net formalism.[iv] To complement this model we propose a graphical representation of the model. We translate as many of the model's features as possible and as faithfully as possible. Furthermore, our visualization accepts a Petri Net state as input, which lets us recreate any execution of the Petri Net model in a 3D model. This supports in understanding the abstract visuals of the Petri Net model.

1. Introduction

Computer science is used to solve problems for other disciplines and create tools to support research. One of the disciplines that benefits from computer science is biology. This thesis proposes a visualization tool usable in the research of *Mycobacteria*.

Though it is generally considered a curable and preventable disease, tuberculosis (TB) still claims over one million lives every year.[i] This makes it the second most lethal infectious agent in the world. To better understand the disease research continues. Much research is done on animal models, these provide valuable insight into the process. One of the animal species used for research is the zebrafish.

The zebrafish, or *Danio rerio*, is a useful addition to the set of species used for TB research. It is not susceptible to the *Mycobacterium tuberculosis* (*Mtb*), the agent responsible for the TB infection in humans. Still, an infectious agent called *Mycobacterium marinum* (*Mm*) is genetically very similar to *Mtb*, and creates similar symptoms in zebrafish.[ii]

Compared to mammalian models based on *Mtb* the zebrafish *in vivo* model has some advantages. Zebrafish are easily bred and kept. Zebrafish are transparent while they are embryos, which makes it possible to easily monitor the state of the infection, without terminating the host. Researchers have even created a genetically modified zebrafish that remains transparent as an adult.[iii] Furthermore, because zebrafish do not develop their adaptive immune system until "several weeks after fertilization"[ii], the innate immune system[1] can be researched in a more isolated manner, revealing a specific part of the reaction of the host. This is particularly interesting because zebrafish have an innate immune system similar to that of humans; its workings are important in understanding the complete set of reactions in zebrafish hosts.

To complement the *in vivo* zebrafish model, researchers of the Leiden Institute for Advanced Computer Science at Leiden University have created an *in silico* model.[iv] This abstraction of the real life *Mm* infection has been implemented in the Petri Net (PN) mathematical formalism. The model follows the *in vivo* model by simulating *Mm* infection and proliferation inside macrophages, phagocytosis, granuloma formation and dissemination. The foundations of this thesis are formed by this PN model along with the paper describing.

In this thesis a visualization for the model, created by Carvalho et al in 2012, is proposed. We take the PN model as input for our visualization. Thus, we reimagine the purely functional, symbolic graphics and data of the PN model to a 3D model. This was our objective for the project.

The main research question of this thesis is how to visualize the biological process of *Mm* infection in zebrafish, as is detailed in the PN model. Attempting to answer this question raises some other questions. Which parts of the PN model are we actually visualizing? Some parts of the PN occur at such a low level in the biology that it is not feasible or sensible to attempt to visualize these. What is the desired balance between a biologically faithful recreation on one hand, and an abstract model on the other? To answer these questions, we keep two things in mind: whether it supports in the understanding of the PN model and whether it keeps the code of the visualization user-friendly.

These research questions have consequences for my approach of the problem. Since the intention is to visualize as much of the PN model as possible, every element of this model is considered. We then

[1] The innate immune system is a basic defence against infections. It is distinguished from the adaptive immune system, which develops later an provides increased protection against numerous encounters, unlike the innate immune system.

look at the importance of each element within the model, and the gain of implementing the element in the visualization. In this way the different elements are evaluated, and the decision to implement or not is made.

By following this method, we discovered the PN model was not intended to be a 3D visualization. We found that certain parts of the PN model were difficult to translate to a 3D visualization, and encountered information that was not complete enough for our purpose. In these cases we looked at the biology as a source of inspiration, and a possible alternative answer when the PN model was unusable. Since the objective of this project was not to model the biological process, biological accuracy is of no importance.

In 2012 Croes has done similar work. He has given us an example on how to visualize the infection and dissemination processes,[v] showing us it is possible. Therefore, our problem is how to visualize the infection process, not determine whether it is possible.

2. Materials and methods

In this chapter we will look at the technologies used for this project. We start with the language used to write the code for this project, after which we look at the materials and methods that are increasingly more specific to our project.

2.1. Language

Our project has been written in the object orientated programming language C++. We used Microsoft's IDE Visual C++ 2013 and its compiler. Visual C++ 2013 generally complies with the C++ 11 standard.[vi] Our software is backwards compatible with Visual C++ 2010; older versions of Visual C++ and other compilers have not been tested. To complement C++, we use several libraries, the most important of which are the graphic libraries.

2.2. Graphics libraries

The main requirements for the graphics solution is the simplicity of both the code. A constraint is the time required to implement our visualization in a certain graphics solution. The goal is to spend as much time as possible on the proper and accurate functioning of the software, not on graphics.

Croes used openGL 4.4 as his graphics solution. For window management and the input of keyboard and mouse he uses freeglut 2.8.1. This is an extension of the openGL library and provides among others many key back end functions. Many of the code structures were already present in Croes' code. Using Croes' code was highly beneficial and saved a lot of time, complying with our constraint. Both openGL and freeglut were found to be easy to use; we did not feel the need to try other solutions.

Croes opted for an object oriented programming (OOP) approach for his thesis, combined with the model-view-controller (MVC)[2] design pattern. We decided to implement both MVC and OOP to make our project as much object oriented as possible. OOP facilitates in making the code easily understood and adaptable; others need to be able to use the code alongside the software itself. OOP modularizes the program into rigidly defined objects. This lets the user digest the code section by section, and provides quicker understanding.

Besides the approach, there are more elements of Croes' software we could use. He created an openGL and glut structure that handled various basic functions. Because both Croes' project and our project deal with a 3D visualization, these structures could be reused. We used this as a code base for our software. Since the objective of Croes' work differed fundamentally from ours, and we did not agree with Croes' implementation choices, the rest his software was discarded in favour of an original implementation. The biggest difference is the input: Croes' software lacked any input, and we used input in the form of a Petri Net model.

2.3. Petri Net and Snoopy

The Petri Net formalism is a modelling language based on graphs. It primarily consists of places and transitions, with arcs between the two. By firing a transition, tokens can move from one place to another. This formalism has applications in a wide range of fields, one of which is modelling biological processes.[vii] The model created by Carvalho et al. is implemented in an extension of the PN formalism, called the Coloured Petri Net.[viii] To run and analyse Petri Nets, a variety of software solutions can be used.

[2] Programming design pattern that separates the interface from the calculations.

To completely visualize all aspects of the Petri Net model, we were required to have complete information of the net, for every time step. None of the software we tried to use provided this. Snoopy came closest to this requirement, by offering files that contain one state of the model. In chapter 4, The Petri Net model, we will see how this is a problem, and how we deal with it. We use files saved by Snoopy[3] as input for our visualization; they are currently the only type of file our software can read. Snoopy is the software Carvalho used to model the PN model in, which means the implementation has already been thoroughly tested. One more reason to use the Spoopy tool is the difficulty to port the model to other tools, caused by greatly differing feature sets. Snoopy was not the only tool we used for our software.

2.4. Other libraries

We had the requirement to read XML files, since the main input data are saved in this format. We chose to include a third party XML parser, called tinyXML, version 2.6.2.[ix] TinyXML enables us to read XML with more flexibility than a proprietary parser could do. In the event of edited XML input or PN model, tinyXML functions reliably. Using a parser also makes it easy to expand the code, should more input from the PN model be desired. TinyXML is lightweight and easy to use, making it more desirable than other parsers.

Another requirement deals with the implementation of position in 3D space. We needed some way to handle calculations of positions. Because these positions are used in various calculations, we thought it preferable to include a third party library. For this project vectors are most suitable, because they can be used for positioning and also for lighting. We have chosen vmath 0.10 by Jan Bartipan,[x] because it is lightweight, easy to use, and contains enough functionality. By using a library for vector calculations, our software becomes easier to understand. Assuming vmath has been thoroughly tested, bugs are less likely. It decreases the number of lines, and separates the math from the implementation.

[3] The Snoopy software uses the XML format to save files.

3. The biological process

Mycobacterium marinum causes infections in zebrafish. In this chapter we will describe how this infection spreads within the zebrafish, from a biological viewpoint. This will help in understanding the next chapters on the PN model and the design.

Macrophages are the main white blood cells with the task to digest pathogens such as Mm.[xi] Once the macrophages have absorbed an *Mm*, several conflicting processes occur. The macrophage begins phagocytosis, with which it intends to digest the *Mm* infection using lysosomes. As is a property of mycobacteria, the *Mm* escapes the lysosomes and survives within the macrophage. The *Mm* keeps itself alive by inhibiting the fusion of phagosomes and lysosomes, impair the identification through inhibiting antigen presentation, inhibiting apoptosis and other antibacterial responses.[xii]

Not only is *Mm* able to survive the harsh conditions within the macrophage, it proliferates in them. Infected macrophages attract uninfected macrophages. Furthermore, the parasitic infection makes the macrophage gain weight. If the macrophage has reached a certain weight it will exit the main blood stream, and “migrate to deeper tissue”.[iv] This migrated macrophage attracts other macrophages; together they form a structure called granuloma.

The granuloma, a typical property of the Mycobacterium infection, is the merged structure of multiple macrophages and other immune cells.[iv] Their boundaries become unclear, and they can be seen as one structure. When the infected macrophage becomes unable to contain the increasing number of bacteria, it bursts. This sends out the bacteria in all directions, infecting the neighbouring tissues.

Some macrophages will be infected and ejected from the granuloma structure. These find their way back into the blood stream, and from that point the process repeats itself. The newly infected macrophage succumbs to its *Mm* infection and eventually forms a new granuloma.

This short description is only a part of the process. Since this is a biological process, one could add processes at a smaller scale repeatedly. Implementing every scale was not realistic for Carvalho’s project, he needed to make choices on what to implement and what not to.(Carvalho 2012) Because we use the model proposed in that paper (hereafter referred to as PN model), most of the abstractions of our software originate from choices made by Carvalho et al. From here I discuss some of the biological features missing in both the PN model and consequently also in our visualization.

3.1 Abstractions

Mm infections are not always obvious, but can be dormant for long periods of time. While macrophages are sufficiently equipped to suppress the infection, they have great trouble removing the infection permanently. Often, the host is asymptomatic, and leaves *Mm* dormant. When the host is weakened, *Mm* becomes active again.[xiii] This behaviour is not included the PN model nor in our graphical representation of the PN. In the abstraction, the infection develops immediately.

To fight any bacterial infection, zebrafish have innate and adaptive immune systems, as is the case in humans. When zebrafish are in the embryo stage, the adaptive immune system is not yet developed. The PN model and our simulation only take the innate immune system into account. Though this is very useful for researching the innate immune system, it makes the PN model and our visualization inapplicable for adult zebrafish research.

Mm does not only infect macrophage cells. Neutrophils are also susceptible, and they play a significant role in the control of the infection.[xiv] Their behaviour has not been implemented in the PN model, nor in our visualization.

All infected macrophages attract other macrophages. In the PN, and in our software, the macrophage only starts to attract other macrophages when it has migrated, and has died.

Time, as with all biological processes, is essential. But the PN model, and our visualization, do not consider time. While the PN formalism does have its own concept of time, this does not correspond with the meaning for the biology. The same can be said about our visualization.

4. The Petri Net model

Petri Nets, introduced in chapter 2.3. Petri Net and Snoopy, are very useful in describing biological processes. By turning knowledge about the biology in to a set of equations and rules one can recreate such a process *in silico*. The advantage of this approach is the ease with which one can analyse the process, change the conditions of the model, and predict the outcome.[iv] Thus the PN is a natural fit for describing a *Mm* infection in zebrafish. We have seen what parts of the biology the PN model does not seek to emulate, but what does the PN model simulate? In this chapter the PN model is analysed.

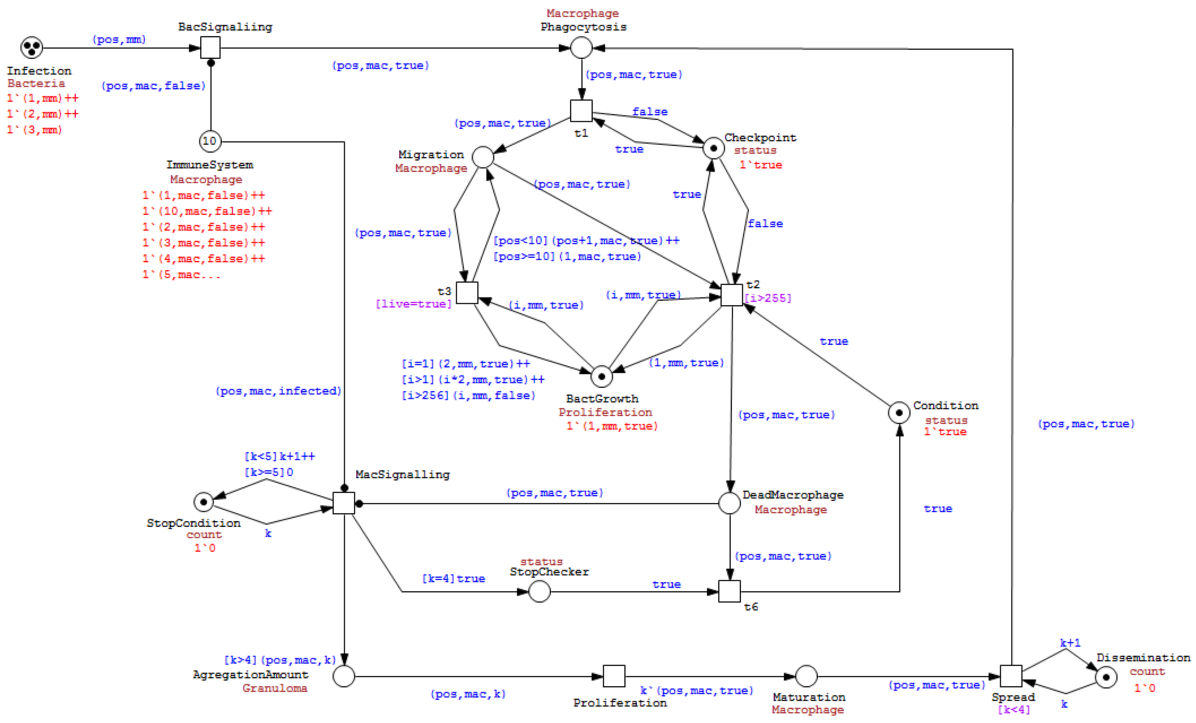


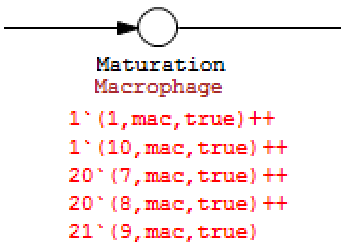
Fig. 1 The zebrafish Petri Net model [iv]

The PN model introduces the concept of relative position. Both the *Mm* and macrophage objects have a position defined by an integer ranging from 1 up to and including 10. The tenth position is linked with the first, forming a circular list. This identifies the object's location relative to the other objects.

The position of the *Mm* is initialized in an arbitrary position in the PN model. It attracts a macrophage from a place called *Immune System*, and starts to proliferate inside it, as described in the biology chapter. Whereas the amount of *Mm* individuals is unspecified in the biology, it has been given a value in the PN. The amount of *Mm* bacteria in a macrophage doubles 8 times, until the proliferation stops at 256 individuals. For every increase in bacteria, the macrophage's position increments by 1, where increments over 10 return the macrophage to position 1. Thus the macrophage always moves 8 boxes before the proliferation stops. Particularly this movement is interesting as it is not directly translatable to a visual equivalent, as we will see in chapter 5 on Design.

After the proliferation process, the macrophage dies according to the PN. A dead macrophage always attracts 4 uninfected macrophages from the *Immune System* place. (The *Immune System* place holds 10 macrophages, one for each position. Macrophages are not removed when they are attracted to

Mm or other macrophages.) The 5 macrophages together form a granuloma, which is then stored in the place called *Maturation*.



```
Maturation
Macrophage
1`(1,mac,true)++
1`(10,mac,true)++
20`(7,mac,true)++
20`(8,mac,true)++
21`(9,mac,true)
```

Fig. 2 The Maturation place

Fig. 2 depicts a screenshot of the *Maturation* place in snoopy. Every line denotes an entry of the place. The first number of a line is the number of macrophages, and the first number within the parenthesis tells us the position of the macrophages. Notice the place holds the number of macrophages, instead of the number of granulomas. Recall that granulomas generally have 5 macrophages. Thus 20 macrophages translate to 4 granulomas. Then why is there only 1 macrophage in position 1?

Not all granulomas behave equally in the PN. We discern two variations: introductory granulomas and non-introductory granulomas. Only the introductory granulomas have been assigned the ability to disseminate macrophages and form new granulomas. The introductory granulomas always cast away 4 macrophages. Implicit is that the dead macrophage is the remaining one. The disseminated macrophages go through the proliferation process in the same way as directly infected macrophages.

As explained earlier in chapter 2.3. Petri Net and Snoopy, the PN model has been created using Snoopy. Snoopy is a powerful tool for editing, visualizing and calculating Petri Nets, but it lets us save only the state that is being displayed. Generally this is the initial state, which in the PN only contains relevant tokens in another place, called *infection*. It does not contain information of the end state (or the dead state as Snoopy calls it), which specifies where the granulomas are in the model. To find this information, we let Snoopy run the Net using *Anim-mode* until it reaches that state. Subsequently *Anim-mode* must to be closed, taking care to use the check the *Always keep marking when closing* box (the *Keep Marking* button seems to be non-functional). Afterwards the Net can be saved. The saved net now contains the end state, which our visualization will use as input. Note that the initial state is lost in the resulting file. Implications for this limitation are described in chapter 7.2 on input.

5. Design

Our visualization uses the Petri Net model as a basis; we attempted to visually recreate the PN model rather than the biological processes themselves. Furthermore, we use the PN model as input, which makes it all the more important to closely follow the PN model. Some of the design has a one to one counterpart in the PN model, and some needed to be translated to create a workable visualization in 3 dimensions. We will now explore the design of our software, and the origin of our design choices by looking at our objects and processes in the order of appearance.

5.1. Zebrafish

The function of the model is symmetrical, that is to say, the head of the zebrafish does not differ functionally from the tail. Still, we wanted the user to be able to distinguish the tail from the head, because it aids in the comprehension of the situation pictured. The host entity is translated to the polygon mesh shown in Fig. 3. The polygon mesh has no other function in our software.

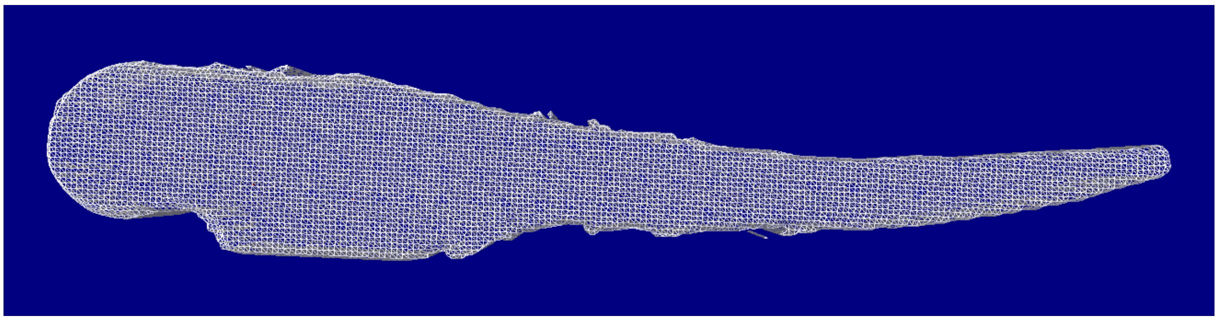


Fig. 3 The zebrafish polygon mesh

5.2. Bounding boxes

To complement the lack of function of the zebrafish model, we included 10 yellow bounding boxes, as displayed in Fig. 4. They are contained within the polygon mesh, and correspond directly with the position concept from the PN model. The boxes are lined up sequentially, so the higher the position in the PN model, the farther back in the fish. This graphical representation means the first divergence from the PN model. In the PN model the first box comes after the last, creating a circular list; in our representation the first and last boxes are not linked together. All of the objects exist within a bounding box.

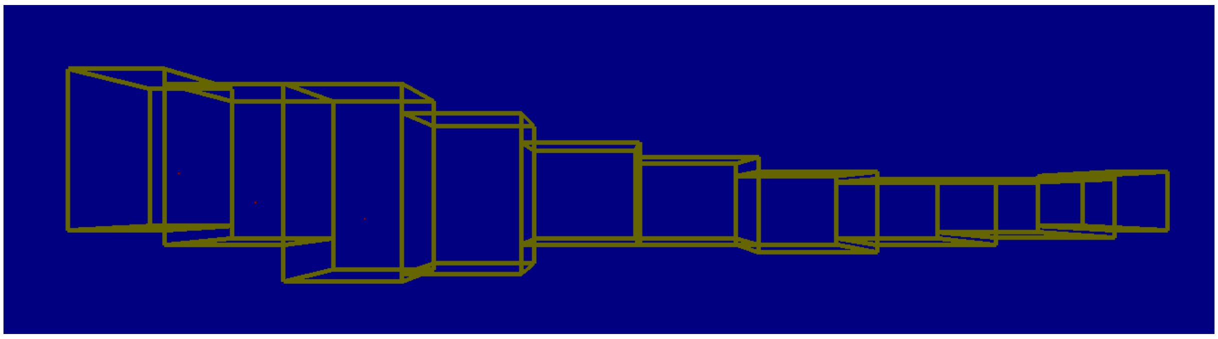


Fig. 4 The 10 segments or boxes of the zebrafish

5.3. Mycobacterium marinum

The *Mm* is visualized by a tiny red dot. Its appearance changes when proliferating, as detailed in chapter 5.4. Macrophages. First we look at the positioning of the *Mm*.

The virtual *Mm* infection starts immediately once the program is running. Though the biology might constrain the possible locations for bacteria injection, we follow the PN model. In the PN model, *Mms* can be in every position. For our visualization, we try to retrace the origin of the infection based on the introductory granulomas. All granulomas that can disseminate have been created by the initial *Mm* injection. And recall that infected macrophages always move 8 positions before migrating. Thus, for every introductory granuloma in the final state, there was an initial bacterium 8 positions prior. This means we know the initial position of the bacteria, if and only if the infection they cause leads to a granuloma that disseminates. Whether all first generation granulomas disseminate is dependent on the settings with which the PN model is run. In the worst case we only know the initial position of one bacterium.

Eventually every *Mm* is captured by a macrophage. From this point on the position (in 3D space) of the bacterium is synchronized to the position of its host. Under no circumstances can it get detached from this macrophage.

5.4. Macrophages

After the *Mms* have been initialized the host's immune system activates. The fish starts to form macrophages from a specified position inspired by biology. The newly initialized macrophage is visualized by a small white sphere. Because the macrophages can be in several conditions, we created different visuals depending on the state of the infection. An infected macrophage is identifiable by the visible red dot representing *Mm* in its centre; along with the proliferation the macrophage turns more red and shows more *Mm* dots; a migrated macrophage dies immediately and its colour switches to black. See Fig. 5 for examples.

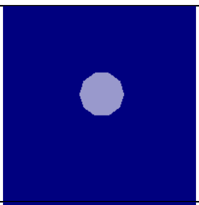
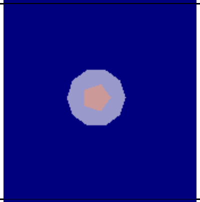
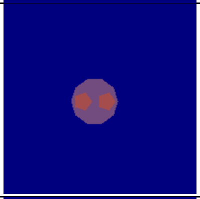
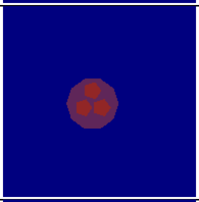
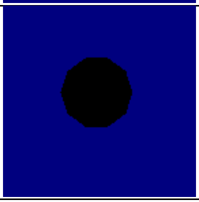
1 Uninfected	
2 Recently infected	
3 Advanced infection	
4 Badly infected	
5 Dead	

Fig. 5 Various visualizations of macrophages according to the level of bacteria proliferation

In the PN model, macrophages that are taken from the *Immune System* are immediately replaced by new ones. The result is that macrophages are continuously present in all boxes. Consequently we discover the requirement that every box must contain a macrophage. Directly translating the PN model would result in unnatural behaviour: macrophages remaining stationary and cloning themselves. Thus we decided to divert from the PN model. Our boxes don't have an infinite stock of macrophages. Instead we implemented a system drawing more from the biology. Our macrophages cycle through each of the 10 boxes, as if they were circulating the blood stream (this will be referred to as the 'circulation' in the future).

Inherent to the nature of circulating blood, two paths were necessary: one in the direction of the tail, and one towards the head. We arbitrarily chose the macrophage that moves in the upper path to move in the direction of the tail. Thus macrophages can be in either the upper part of a bounding box or the lower part. Since the macrophages continuously move through the boxes, each bounding box contains or soon will contain a macrophage. This could slightly alter the order of infections, but time was never part of the PN model, so there is no conflict. In this manner we have a functional equivalent of the PN model.

Because the number of macrophages is finite in the visualization, it decreases when macrophages migrate to deeper tissue, or are attracted to previously migrated macrophages. To prevent the depletion of macrophages creating conflicts with the PN model, the zebrafish introduces a new macrophage for each that exits the circulation. This behaviour has been inspired by the biology, where the infection triggers the release of new macrophages.[xv]

In the PN model the macrophages always move 8 times before they migrate to deeper tissue and form a granuloma. But the shift from the PN models' position system to the creates a conflict: 8 steps in the PN model does not necessarily lead the macrophage to the same box as 8 steps in our visualization. To keep the end result of our visualization in line with the PN model's final state, we chose to take the position of the granuloma as the macrophage's target. Consequently in our visualization the macrophages will not necessarily move 8 boxes, but migrate to deeper tissue the moment they've reached the target box, as determined by the PN input.

5.5. Proliferation

In the PN model, along with every position change in the PN model, the number of *Mm* bacteria doubles. Should the macrophage migrate before having travelled 8 boxes, the number of *Mm* bacteria will not reach 256. Since this is not relevant for the result of the visualization, we decided to keep this inconsistency. Along with the proliferation, the macrophages' weight increases and movement speed decreases slightly. This will let other macrophages, that all move at the same speed, overtake the infected macrophage. We decided to minimize this effect, so the user would not feel hindered by the delay. This is not a feature of the PN model, and has been inspired by the biology. After the migration of the macrophage the formation of the granuloma begins.

5.6. Granulomas

As can be seen in Fig. 6 the granuloma is represented by a yellow sphere. It is formed when a fifth macrophage joins an existing cluster of macrophages. A requirement determined by the PN model is the distinction between introductory and non-introductory granulomas. If the granuloma that forms is an introductory granuloma, the dead macrophage will burst, sending out bacteria to the surrounding macrophages. The other macrophages will get infected and disseminate. The remains of the granuloma, containing just the dead macrophage, remain visible, analogous to the single macrophage that stays in the PN model's *Maturation* place.

The non-introductory granulomas do not disseminate. They remain filled with 5 macrophages for the duration of the simulation, exactly like in the PN model.

5.7. Iteration

The released macrophages return to the circulation, and the *Mm* starts proliferating. This is analogous to the change of the macrophage's position in the PN model. The macrophage circulates as long as necessary to reach the box in which it will form a granuloma, and form a granuloma the same way the introductory granuloma was formed. The difference in these second generation granulomas is that they are always non-introductory. As is evident in the set of 5 macrophages representing such a granuloma in the *Maturation* place, they are not capable of dissemination.

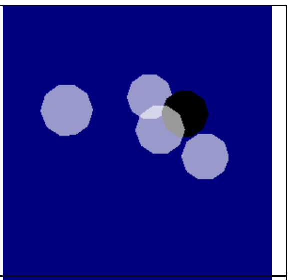
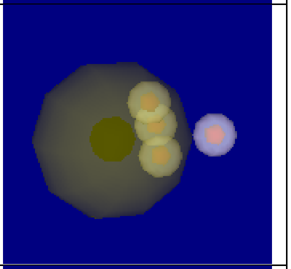
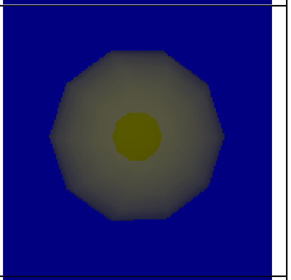
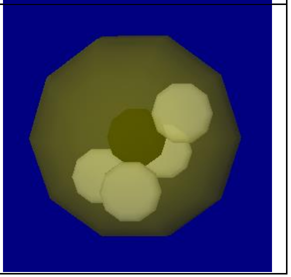
1 Formation	
2 Dissemination	
3 Introductory	
4 Non-introductory	

Fig. 6 Various visualizations concerning the granuloma

5.8. Class structure

Our program contains the following classes.

1. Zebrafish
2. Bounding Box
3. Sub box
4. *Mm*
5. Macrophage
6. Granuloma
7. PN

The relations between each class become clear once we look at the diagram of Fig. 7. Granulomas contain up to 5 macrophages, and macrophages contain up to 1 *Mm* object. This is different from the visual representation. Programmatically just one *Mm* object exists within a macrophage, but each *Mm* is visualized by up to three dots. The number of dots is determined by the integer *bacteriaProliferation* of the macrophage class. *Mm* do not need to be part of a macrophage, nor do macrophages need to be part of a granuloma. The zebrafish, of which there exists just one, has 10 bounding boxes. Each bounding box has two sub boxes, to distinguish between the upper and lower parts of the box. Both macrophages and bacteria have pointers to the sub boxes that signify their positions. The list of attributes and operations is by no means complete, but primarily means to illustrate the general structure of the code. In the rest of this chapter we will look at some key aspects of the implementation.

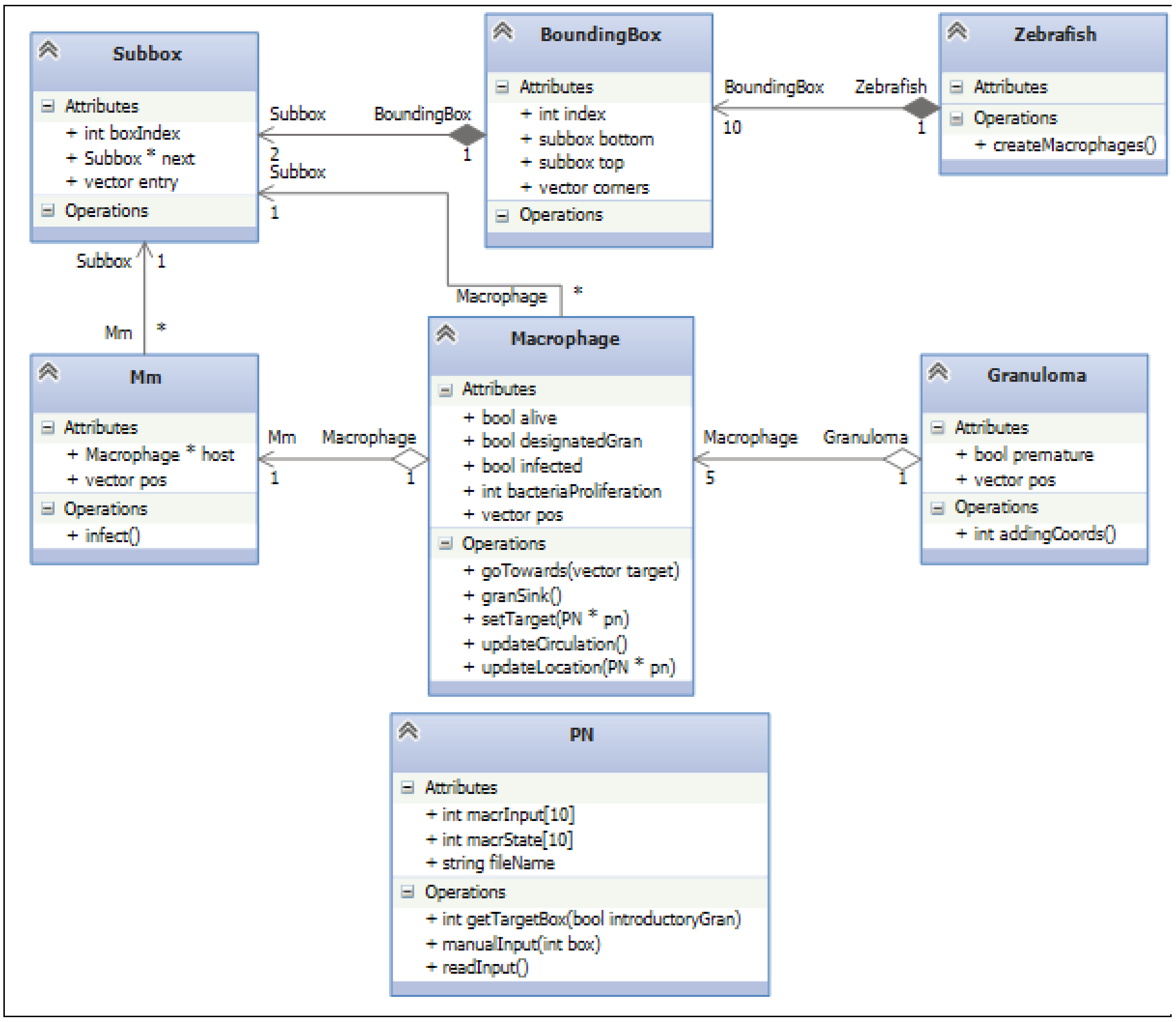


Fig. 7 Class structure

Two sub boxes complement the function of the bounding box. Where the bounding box structure is the translation of the position property in the PN model, the sub boxes facilitate the functioning of the circulation.

Part of this class structure has been created by Croes for his 2012 thesis. He used the granuloma, *Mm,* macrophage, zebrafish and bounding box classes. Croes created the visual representation of the zebrafish: the bounding box class with the 10-segment layout, complemented by the polygon mesh. He created the object classes (bacteria, macrophage and granuloma), and the visuals used for each object. The implementation of the classes of Croes' project could not be used, and original solutions had to be created.

6. Implementation

In this chapter we will look at how we have implemented the design described in chapter 6 on Design. We will not describe trivial parts of the implementation, and have made a selection of several parts that were problematic or difficult. First we describe the process that determines the position of the macrophage, then the input and output, and finally the interface of the software.

6.1. The *updateLocation* function of the macrophage class

This function captures the most essential behaviour of our visualization, the movement of the macrophages. For that reason, we discuss it in more detail than other aspects, using the pseudo code in Fig. 8. The function is divided into two sections separated by an if statement, and preceded by two calls to other functions.

```
void Macrophage::updateLocation(PN *pn){
      setTarget(pn);
      granSink();
      if (there is no target){
            if (the bacteria have proliferated maximally){
                  //turn the infection dormant
                  infected = false;
                  bacteriaProliferation = 1;
                  movementSpeed = 100;
            }
            updateCirculation();
      }
      else{                              //There is a target
            if ((!designatedGran || bacteriaProliferation > 0) &&
            subbox->boxIndex == targetBox){
                  goTowards(target);
            }
            else{
                  updateCirculation();
            }
      }
}
```

Fig. 8 Pseudo code for the updateLocation function

These first calls set the macrophage's target (we define target as: the granuloma the macrophage will go towards, along with the specific location in 3D space within the boundaries of the granuloma). They do so for two very specific situations.

1) *setTarget* sets the target for macrophages infected by bacteria. This function is also divided in two subsections. One sets the target for macrophage infected by loose (injected) bacteria. The other sets the target for disseminating macrophages. The target is always determined by calling the *getMacrTarget* function of the Petri Net parameter.
2) *granSink* sets the targets when the macrophage is uninfected, and it is attracted to a granuloma in the same box.

With all possible ways for macrophage to get a target covered, we can look at the rest of the function. If these functions have yielded no target, the macrophage will follow the circulation for now, with one exception. If the bacteria inside the macrophage have proliferated maximally, the infection will turn dormant. This has been implemented as a fail-safe for macrophages that are not given a target, and it gives to the user extra flexibility when manually inputting targets. This dormancy has been inspired by biology, but is not intended to be an accurate reflection of the biology involved.

If *setTarget* or *granSink* did yield a target, the macrophage circulates until it is in the same box as the target, at which point it will go towards the target. It will not be able to go towards a target immediately after infection (*bacteriaProliferation > 0*), to give the user the image that the macrophage needs to return to the circulation before disseminating, as happens in the PN model. *GoTowards* will be called repeatedly from the first change of direction, through the arrival at the target, until the macrophage returns to circulation.

The circulation is in reality a set of 20 predetermined points, along which circulating macrophages travel. These points are called entry points, and belong to a sub box. The entry point is derived from the dimensions of the bounding box of which the sub box is a part. Macrophages travel from the entry point of one box to the entry point of the next. A pointer structure determines the order of the sub boxes, and with it the direction of the circulation. In the upper sub box of a bounding box, the macrophages travel towards the tail. In the final bounding box that represents the tip of the tail, the macrophage switches from the upper sub box to the lower sub box. Then it returns in the other direction, towards the head.

Macrophages only leave this circulation if their target is set. The macrophage's target in 3D space is determined by the granuloma they are going to be a part of. The granuloma picks a random spot within its circumference. We initialize the granuloma the moment the target of a macrophage is set. Granulomas start to attract other macrophages after the first macrophage arrives at its target, and become visible when the fifth arrives. The granuloma becoming visible corresponds with the formation of the granuloma in the PN model. The extremely early initialization of granulomas is not a part of the PN model, nor of the biology. The reason for this is that we need to reserve a space for the granuloma to occupy, within the box.

6.2. Input

We included separate modes for input from PN model files and manual input by the user. Reading data from a file is the main functionality. The Snoopy software creates these files in XML format, and they contain all information we use in the visualization before the visualization starts. Alternatively the user can decide to manually input the data normally found in the files. We decided to include this because of testing purposes, and because it gives the user extra flexibility in getting the desired visualization. Both input modes use the same pathways for expressing the input.

The PN class represents a single state of the model, with the data saved in the *macrInput* array (see Fig. 7 Class structure). This array represents the data in the same way the PN model does: as the number of macrophages per position. Another array part of this class, *macrState,* array holds the current state of the visualization, in the same terms.

Infected macrophages ask the currently used PN object for a target. The *getTargetBox* function determines where to form the new granuloma. The return value of this function is determined by two aspects: the difference between the *macrInput* and *macrState* arrays, and secondly the *introductoryGran* parameter that indicates whether to fetch an introductory granuloma or a non-introductory granuloma. Every successful fetch updates the *macrState* array, so each desired granuloma is created once.

In the case of manual input, the user chooses in what box to form a granuloma through the numeric keys. Any numeric input will fill the *macrInput* array appropriately, using the *manualInput* function. There are several functional differences as opposed to the reading of the PN file. One is the inability to set the amount of introductory granulomas. This information is not available without full knowledge of the PN. For manual input this amount is by default set to 3. The other is the extra flexibility described previously. The user can choose to postpone the input of subsequent numerals.

The desired position of bacteria is derived from the *macrState* array, and functions in a structure similar to that for macrophages. Because bacteria are immediately initialized, the macrophage data must be available from the start of the program. Because this is not the case in manual input, the position of bacteria is unrelated to the workings of the PN model.

6.3. Output

The software outputs two windows. One provides a log of the program run, which contain significant information that is useful for debugging purposes. The other contains the actual visualization.

The visualization window shows the zebrafish model, and the bounding boxes, that roughly follow the edges of the model. To show all our features and make the effect of the input clear to the user, both are displayed by default. This can be changed any time during execution, as will become clear in the next paragraph on the interface.

6.4. Interface

To provide the user even more flexibility in the usage of the software, we implemented some auxiliary functions, called by specific keyboard keys and the mouse. The interface buttons were chosen to intuitively make sense. Unfortunately, this was not possible for all functions. Some of these functions are implemented for user convenience, and some can change the look of the visualization.

Various elements of the graphical presentation of the program can be changed by the user during runtime. Flat shading has been implemented to make the fish appear to have physical substance. The opacity of the model can be changed by the user, by using the '<' and '>' keys. The inside of the fish can be rendered invisible by maximizing the opacity. Alternatively one can turn off drawing the model by minimizing the opacity. This is useful for less powerful machines. The drawing of the bounding boxes can also be turned off by the user, for a cleaner visualization.

This is the full set of interface keys with their effect.

- The 'p' button pauses the flow of the macrophages.
- The 'r' button resets the software: clears all objects so the user can start over.
- The 'z' button resets the speed of the visualization to 1; the 'x', 'c', 'v' buttons set the speed to respectively 2, 5 and 10.
- The 'b' button toggles the drawing of the bounding boxes.
- The '<' button decreases, and the '>' button increases the transparency of the zebrafish polygon mesh by 10 percent.
- The 'w', 'a', 's', 'd' buttons and holding the left mouse button each play a part in moving the camera.
- The scroll wheel of the mouse moves the camera in the direction it is facing.
- The numeric keys are used for manual input, as described earlier in this paragraph.
- Numeric keys set the input mode to manual, and inputs the key pressed.

8. Improvements over Croes' research

There are many similarities between Croes' thesis and this thesis. Both visualize the infection of *Mm* in zebrafish. Both use a 3D visualization, and both use the same assets. There are also several significant differences.

When first studying his work, a encountered several problems with his solution. The keyword that describes his visualization best is random. The z coordinate of the injected bacterium is generated randomly; the granuloma that disseminates is chosen randomly; the location of granulomas is not even specified, though presumably this is random too. Though Croes used the same amount of bounding box, they were not used to specify the position of the objects. The result is a program of which the outcome cannot be determined beforehand. It at best visualizes one possible instance of the *Mm* infection. This limits us in the practical applications of the visualization.

As opposed to Croes' software, our software always behaves predictably. Our visualization functions with (but is not dependent on) the input of a PN save file that has been run to the final state. The location of the granulomas (and macrophages), relative to the position of other objects, is determined beforehand by the PN model. By editing the PN model, the user can change the visualization as desired.

Researchers who use the PN model are not restricted to the abstract visuals of the PN software, Snoopy. This leads them to a better understanding of the results of the PN, and of any changes they make to the PN model. This is not possible in Croes' visualization.

9. Conclusions and discussion

The visualization we proposed reads a Snoopy file that contains the result of the PN model. To date it is not yet possible to ability to read all states of a PN model, therefore we decided to read only the final state. This contains all the information on the presence and location of granulomas, including how many macrophages they contain. From this information, we can retrace the origin of some of the bacteria that infected the macrophages. Using the read end state, and calculated begin state, we interpolate the behaviour in between macrophage infection and the end result. This makes our visualization a generally accurate graphical representation of the PN model.

The underlying model differs in several non-critical points. The position system in the PN model has not retained the same ordering in our visualization. The position of bacteria can currently not be read by our software. We can reliably retrace the information, but only when the file has been created using the preferred setting.

We separated the visual element of the PN model from the calculations and data. Our software interprets and translates this visual element. The data of the PN model often inhibits comprehension of the process. Biologists researching the process are not trained to comprehend mathematical formalisms, and even graphical ones such as the Petri Net can be difficult to read. Hence, using our visualization, these biologists' understanding of the PN model is simplified.

Another application of our visualization is its use in demonstrations. The function of the PN model can be explained through the use of a live demonstration or a video (with 3rd party recording software) of the program execution. By changing the marking of the PN input, the user can exactly visualize the desired process. If the Snoopy file input doesn't produce a visualization that suits the user's needs, manual input provides even more flexibility. Additionally the user can change visual options during runtime, and easily modify the code if necessary. The result is unprecedented flexibility for a visualization of *Mm* infection in zebrafish.

This, complemented by the option of manual input gives the user a fully customizable visualization. This in contrast with Croes' software, that produced one random visualization that could not be customized to fit the user's preferences. The flexibility makes our software much more useful in the scenario's we described. This success is attributable to using the PN model as input for our visualization.

10. Further research

For this particular PN model, the visualization is generally accurate. But should the PN model be expanded upon or otherwise changed, the assumptions we make, and the information we derive from the final state could no longer hold true. The ensuing loss of accuracy is already the case if the preferred settings in Snoopy are not chosen.

Therefore, we think it would still be desirable to implement what we did not yet accomplish: to have the ability to read all states of the PN model. It would greatly increase the usefulness of the software, and make the software future-proof by being more flexible with the PN model readout. Because of the importance of this future work, we took this expansion into account while writing the code. This and the OOP approach to the problem will make it relatively easy to adapt the software when the need arises. A PN class has been set up, the objects of which define a single state in the PN model. One would only have to cycle through the states of the PN model, and the visualization will update accordingly.

Carvalho's current, unpublished research is to expand the PN model we used for our visualization.[xvi] It delves deeper into biological processes the 2012 paper only considers superficially, if at all. Many of these sub-processes deal with specific substances such as calcium or low level processes like apoptosis inhibition. Expanding the visualization to include these elements is probably not realistic, but it is worth researching whether there are other parts of the expanded model we can implement.

11. References

[i] World Health Organisation. "Tuberculosis". http://www.who.int/mediacentre/factsheets/fs104/en/. Last accessed at 18 Mar. 14.
[ii] A. H. Meijer, A. M. van der Sar, C. Cunha, G. E.M. Lamers, M. A. Laplante, Hiroshi K., W. Bitter, T. S. Becker, H. P. Spaink. "Identification and real-time imaging of a myc-expressing neutrophil population involved in inflammation and mycobacterial granuloma formation in zebrafish." *Developmental and Comparative Immunology*, vol. 32, no. 1, pp. 36–49, 2008.
[iii] R. M. White, A. Sessa, C. Burke, T. Bowman, J. LeBlanc, C. Ceol, C. Bourque, M. Dovey, W. Goessling, C. Erter Burns, and L. I. Zon. "Transparent adult zebrafish as a tool for in vivo transplantation analysis" *Cell Stem Cell*. Vol. 2, no. 2, pp 183-9. 2008.
[iv] R. V. Carvalho, J. Kleijn, A. H.Meijer and F. J. Verbeek. "Modeling Innate Immune Response to Early Mycobacterium Infection". *Computational and Mathematical Methods in Medicine.* Vol. 2012. 2012
[v] X. Croes. "3D Visualization of Mycobacterium infection on zebrafish". Internal report. 2012
[vi] Microsoft. "Support For C++11 Features". http://msdn.microsoft.com/en-us/library/hh567368.aspx. Last accessed at 18 Mar. 14.
[vii] F. Liu and M. Heiner, "Colored Petri nets to model and simulate biological systems," *CEUR Workshop.* Vol. 827, pp. 71–85. 2010.
[viii] K. Jensen and L. M. Kristensen, L. Wells. "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems". *International Journal on Software Tools for Technology Transfer.* Vol. 9(3), pp.213-254. 2007.
[ix] L. Thomason. http://www.grinninglizard.com/tinyxmldocs/index.html. . Last accessed at 18 Mar. 14.
[x]J. Bartipan. http://bartipan.net/vmath/doc/. Last accessed at 18 Mar. 14.
[xi] D.O. Adams, T.A. Hamilton. "Molecular basis of macrophage activation: diversity and origin" from "The Macrophage", *Oxford University Press*, pp. 75–114. 1992.
[xii] E.N. Houben, L. Nguyen, J. Pieters. "Interaction of pathogenic mycobacteria with the host immune system." *Curr. Opin. Microbiol.*, Vol. 9, pp. 76–85. 2009.
[xiii] V. Sundaramurthy, J. Pieters. "Interactions of pathogenic mycobacteria with host macrophages." *Microbes and Infection.* Vol. 9. pp 1671-9. 2007.
[xiv] C. T. Yang, C. J. Cambier, J. M. Davis, C. J. Hall, P. S. Crosier, and L. Ramakrishnan. "Neutrophils exert protection in the early tuberculous granuloma by oxidative killing of mycobacteria phagocytosed from infected macrophages." *Cell Host & Microbe*, vol. 12, no. 3, pp. 301–312. 2012.
[xv] Personal communication with dr. F. Verbeek.
[xvi] Personal communication with R. Carvalho.