Chapter 6

Dynamic Biological Visualization - 3D Modeling using WebGL

INTRODUCTION

During the initial implementation the *Streptomyces* morphological model presented in Chapter 6, the rational choice of tool for the task at hand was MATLAB®. A powerful, high-level, yet easy to use programming language, MATLAB is commonly used by scientists and engineers alike for technical computing. Using an object-oriented approach, the *Streptomyces* mycelium was modeled as a collection of branches, which in turn were modeled as a collection of points with various properties. Properties included whether a point was the start of a new branch or a cross wall location, and its compartment type (apical, subapical or hyphal). Point positions (x, y, z coordinates in 3D space) and properties (denoted in binary notation) were stored within matrices. During the simulation, the processes of growth, branching, cross wall formation, as well as fragmentation, were looped through sequentially, resulting in change of the morphology, and consequently, change in local oxygen concentration. The assumption was that continued growth and branching would deplete oxygen within a pellet, while increasing pellet density, affecting diffusion. Decreased local oxygen concentration would in turn affect growth and branching, resulting in a feed-forward, feed-back mechanism of pellet formation, as well as differentiation, which is known to correlate to productivity. Simulation results were matrices of branch points, reflecting the morphology resulting from a particular set of input parameters.

Matlab functionality allows plotting of two- or three-dimensional plots for data visualization. In this way, if the points of each branch in the mycelium were connected by lines, a simple visual representation of the mycelium could be obtained. To improve upon this still rather crude output, the freely available Persistance of Vision Ray-Tracing software (www.povray.org) was used to create more visually realistic images of simulated pellets (see Chapter 6). Pov-Ray uses ray tracing for image generation, which is a technique that involves tracing the path from a light source through pixels in an image plane and simulating the effects that this light beam would have as it encounters the virtual objects in its path. Although a high degree of visual realism could be achieved, Pov-Ray suffers from lack of dynamics, and individually rendered images need to be stitched together to generate a movie or simulation. Although the Pov-Ray output allowed us to demonstrate that model results provided a realistic portrait of mycelial morphology, this form of visualization was time-consuming and not intuitive.

To develop an interactive version of the model, and drawing inspiration from the Bio-digital Human project (https://www.biodigitalhuman.com/home/) and the Open Worm Browser (http://browser.openworm.org/), part of the code was converted to Javascript so

as to generate a web-based 3D version of the model of *Streptomyces* pellet growth and morphological development. The idea was to create a dynamic model in the browser, where a user can see a pellet grow while adjusting parameters and immediately see the effect of parameter changes on pellet morphology, and ultimately, production.

Use of the Web as a software deployment platform has only recently become possible with the emergence of the HTML5 (http://www.w3.org/TR/html5) and WebGL (http://www.khronos.org/webgl) standards. HTML5 complements capabilities of the existing HTML standard in order to enable support of the latest multimedia, such as the embedding of audio and video directly into web pages, effectively to help in the transformation of the browser into a programming environment (Anttonen, Salminen *et al.* 2011). To give an example, the canvas element, a 2D graphics API (application programming interface) for defining shapes and bitmaps rendered directly in the web browser, is a new feature of HTML5, while WebGL enables visualization of Graphics Processing Unit (GPU) hardware-accelerated 3D graphics in the browser without additional software, plug-ins or extensions. With the help of THREE.js, a lightweight, cross-browser Javascript library, a simplified version of the model could be reprogrammed as a WebGL 3D computer graphics element, able to run in the browser.

Programming of the model in Javascript involved re-thinking the way that the model is structured and coded. Javascript, or JS, is an interpreted language, originally implemented for web browsers to enable user interaction and change of document content via client-side control of the browser. The power, or rather difficulty of Javascipt, lies in the fact that it is weakly (or loosely) typed, meaning that variables do not have a type (ie. int or char), and can hold any object, without restrictions. In order to maintain the hierarchical structure of the model (a mycelium or pellet containing branches, which in turn contain points), an object-oriented approach was used in Javascript, which involved creating different functions for each hierarchical level, with corresponding properties.

During initialization of the page, a THREE.js scene is constructed, including the mycel (short for mycelium) object itself, consisting of branches, with associated branchunits (lines), as well as a camera, light source, and renderer. Application interaction is possible by using the mouse to rotate the pellet during growth and to zoom using the mouse wheel. One javascript function file contains all the variables involved, to be called from the other files as needed. A graphical user-interface (GUI) is provided with, for the moment, four variables: branching interval, tip growth rate, and tip growth angles phi and theta.

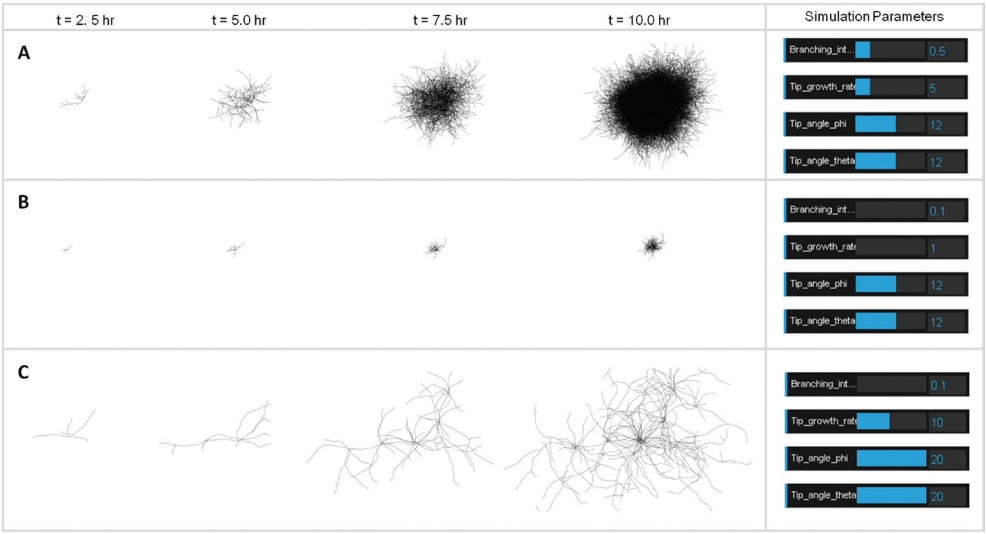

Figure 1. *WebGL simulated morphologies. (A) Dense pellet formation. (B) A very small pellet results when the tip growth rate is decreased. In this simulation, branching interval was also set to 1 branch per 10 µm, but due to the very slow growth rate, the effect this has on morphology is not evident. (C) A mycelial mat structure results from a low branching interval and high growth rate. Tip angle variation was also increased in this simulation, resulting in hyphae which curl more during growth. Parameters are provided in the graphic user interface boxes to the right.*

Adjusting the variables results in the development of various morphologies. A comparison between three different simulation runs with different parameter initializations is shown in Figure 1. The default parameters result in a dense pellet morphology (Figure 1A), while decreasing the growth rate demonstrates that in adverse conditions, small pellets may develop (Figure 1B). Increasing the tip growth rate as well as the interval between branches results in a mycelial mat morphology (Figure 1C). The simulations are a good illustration of the large difference in morphology that can result when only a few parameters are adjusted. Moreover, improving upon the MATLAB model, parameters can be changed during the simulation, illustrating how an influx of nutrients, or a change in fermentation parameters can potentially affect morphology. Figure 2 provides a screenshot from a simulation run during which parameters were changed dynamically. The resulting pellet is dense on the inside, but with a more open outgrowth of external branches.

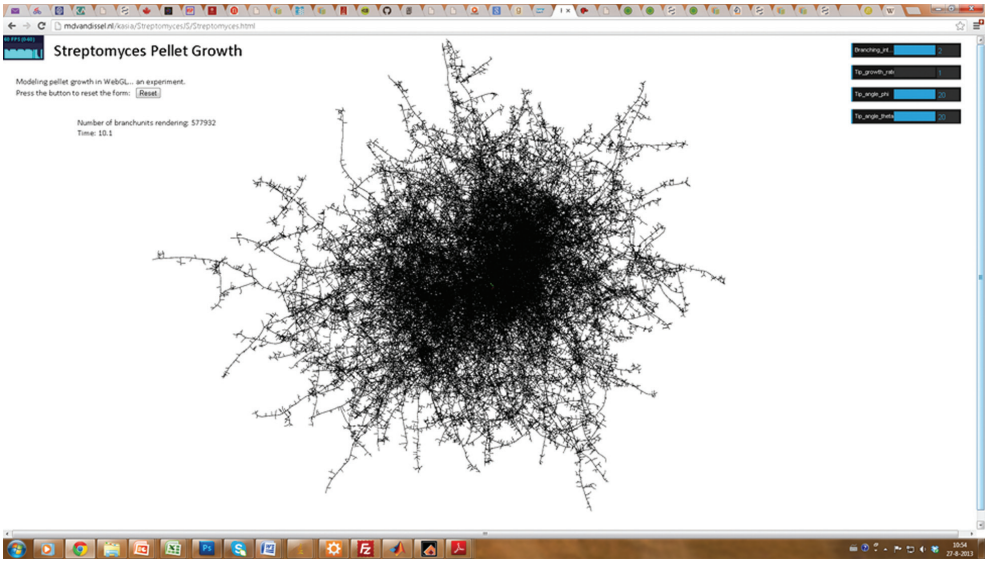

Figure 2. *Screenshot from simulation run, demonstrating the effect of parameter changes on morphology. An initially fast tip growth rate, and average branching interval resulted in rapid growth of the pellet. Branching interval was then decreased, resulting in larger distances between branches, and enabling extension of long branches outside of the pellet core. Near the end of the simulation, tip growth rate was set at a minimum, while branching interval was increased, resulting in short distances between the new branches emerging on the long outreaching hyphae.*

FUTURE PERSPECTIVES

At the moment the WebGL simulation is a simplified version of what was previously coded in and achieved with Matlab and PovRay. The basis, however, is there. With some additional steps, the code can be improved and extended to provide a more realistic and useful morphological simulation.

Firstly, it is necessary to implement the oxygen diffusion algorithms to model oxygen diffusion and consumption by the growing pellet. Solving equations in the browser is something that is now possible to achieve in WebGL by using the graphics processor to simultaneously solve sets of parallel equations (Janik 2012). This is currently only possible for sets of linear equations, but developers are working on creating non-linear equation solvers for the browser. Secondly, collision detection should be implemented to prevent hyphae from overlapping within the centre of the pellet. Because of the large interest in the use of WebGL to create 3D games in the browser, numerous algorithms exist to implement collision detection. Thirdly, it would be beneficial to implement a pellet render function once growth

has completed to model the lines as three dimensional objects. Since immediate modeling of the hyphae as three-dimensional cylinders or tubes is very taxing on the browser, lines were chosen to visually represent the mycelium and give a realistic approximation of the morphology, albeit only at lower zoom levels. With the help of a pellet render function, during zooming and morphological analysis of the pellet, a user would be able to better visualize and analyze the pellet packing and calculate the void fractions corresponding to various modeling scenarios.

As the web technology for in-browser modeling improves, the power and speed of the simulation will also increase, facilitating model expansion and improvement. Adding and implementing new experimental data from fermentation trials based on different strains and morphologies, as well as gene expression data relating to morphogenesis, such as the effect of ssgA overexpression (van Wezel, Krabben *et al.* 2006), will increase the biological relevance of the model and may lead to new insights. In the future, such a dynamic, easily-accessible, 3D-modeling approach may prove to be valuable for strain improvement, with modeling effectively a test-drive for the fermentation process to pre-assess the effect of variables on productivity (Celler *et al*. 2012). At present, it demonstrates the application of WebGL, a new web technology for three-dimensional modeling in the browser, for visualization and modeling of *Streptomyces* pellet growth. The tool can be used for teaching purposes to showcase the mathematical beauty and complexity of *Streptomyces sp.*, and the intricacies of branching growth and morphogenesis in liquid cultures.