Modeling Infiltration
Bachelor Project

by Ramon Wenger
at SCG
http://scg.unibe.ch
University of Bern

Abstract

The goal of the Bachelor project **Modeling Infiltration** was to create a dynamic graphic representation of the process of water infiltrating any kind of soil over time (the process in short called *infiltration* throughout the documentation), the implementation of which is realized in Javascript.
The purpose of this document is to introduce the project, lead through some extracts of the code and explain its purpose. It provides the reader with the ability to read and understand the code that is the basis for the application.

Acknowledgements

I'd like to thank Professor Oscar Nierstrasz for providing me with guidance and helpful tips when I was at my wit's end.
I also want to thank Professor Peter Germann for keeping his patience every time he had to explain his formulas and their implications to me.
Last but not least my thanks go out to my friends and especially to my girlfriend Ramona for all the times they were understanding when they had to do without me.

Table of contents

Introduction

The basis of this project was the desire of Professor Peter Germann of the Institute of Geography, University of Bern, to assist readers of his book in understanding the models within, via a web application. The book in question contains Prof. Germann's research about the behavior of water when infiltrating the soil during rain periods.
This software engineering project was conducted with a client (P. Germann) and a budget (as time in week increments), similar in fashion to a SE project in the "real world".

Assignment

The goal of the assignment was to have a web application, which being input an array of water pulse data would produce a graph. This graph would show the maximum depth reached by the water infiltrating the ground. The formulas for handling the input came from the aforementioned research, and had to be translated from mathematical formulas to functions in a programming environment.

The basic draft of the requirements contained possibilities for the users of the application to input their own water pulse data, ranging from a text input to a manipulation of the graphic representation of the pulses via drag & drop. This premise had to be given up though, after discovery of certain behaviors of particular types of input, which could not be handled in a dynamic fashion, and no general rule for valid versus invalid input values could be found.
The final requirements contained different sets of input arrays for which the application would produce valid outputs.

Further, the illustration of the graph and the pulses was to be animated, to see the build up of the two parts of the illustration over time.

The application was agreed to be done in Javascript and HTML.

HTML structure and element properties

Doctype

The doctype in an HTML document servers as an instruction for the browser on how to handle the content found in it.
The doctype chosen for this project is

```
<!DOCTYPE html>
```

which stands for the new standard, HTML 5.
HTML 5 is used because it is the current state of the art at the moment of writing, and also because one HTML 5 elements serves an important part in the application.

Canvas

The *canvas* element is a new element in HTML 5 and can be used to generate dynamic images via Javascript.

```
<canvas id="canvas-pulses" width="800" height="200"></canvas>
```

The *canvas* can be styled via CSS as most other elements.

```
background-image: url('raster.gif');
background-position: top right;
```

Here both of the *canvases* have been given a raster as a background to have a relation as to the dimensions of the illustration.

The rest

The rest of the HTML elements consists of simple *buttons*, *divs* and *images*. Most of them only have their standard attributes, as well as an id so to be able to be identified by the CSS and Javascript. All their functionality is handled by the Javascript.

Javascript Code

Note: At the start of the project, the question posed itself whether to use a framework like jQuery or anything similar. It was decided against using frameworks, to explore the native functionality and the ups and downs of Javascript

Logic

Equations and intersections

One of the challenges was to solve equations, as the Javascript code has no native functionality in that regard. Specifically, the goal was solve equations between linear functions, and ones in the form of linear functions and root functions.

The solution for this challenge was to adapt Newton's method.

> The idea of the method is as follows: one starts with an initial guess which is reasonably close to the true root, then the function is approximated by its tangent line (which can be computed using the tools of calculus), and one computes the *x*-intercept of this tangent line (which is easily done with elementary algebra). This *x*-intercept will typically be a better approximation to the function's root than the original guess, and the method can be iterated.
> [1] Newton's method on Wikipedia

The adaption of Newton's method used in the project looks for the intersection of two functions, which both form the first half the parameters of the function, together with a minimum and a maximum value, between which the intersection is supposed to be found.
The intersection-function compares the two functions' values at the minimum point, and returns the minimum when they match to a certain predefined accuracy.
If they don't match, it compares the values at the maximum, and returns if they match to the aforementioned accuracy.
If they don't match either, the function checks the validity of the interval between the minimum- and maximum parameters, and returns a *false* value if the two functions values can't intersect in between.
Finally, the function halves the interval between minimum and maximum, and calls itself again, with the first half of the original interval at first, and the second half if the first returned a *false* value.

The advantage of this solution is the performance, as the iterations needed to find a solution are much less than an approach of iterating through every possible value of x, for example.

Objects

The main object used in the software design is the *Pulse* object.

The *Pulse* object is a representation of a water pulse, and its 3 main properties are the time of its start, the time of its end, and its intensity. Based on these 3 values and some specific formulas the rest of its properties are calculated.

A pulse has a function each for the so called wetting front and the draining front, named *wettingfront* and *drainagefront*/*drainagefrontStart*, respectively. They are linear functions and describe the depth of the specific front at any given time of the simulation.
The wetting front is used when the current pulse is the furthest down of all pulses so far, which in our simulation means it is the currently active pulse.

The draining front is used at any other point, to determine the speed of the descent of the water. For the use in this specific application we have one draining front calculated from the beginning of the pulse, and one from the end of it.
These two fronts meet at an intersection point, computed at creation of the object, via the aforementioned intersection.
From this intersection point on, the function *combined* describes the descent of the water without any more new water input like before the intersection. What this means is the function is not linear, but behaves like a root function.
Each pulse can have a *jump*-function, which depends on the next pulse in line, and is determined at runtime. If the next pulse is smaller in intensity than the current one, the current pulse's *jump* function just returns null. If on the other hand the next function is larger, however, the *jump* is a linear function, describing an increase in speed from the current function's draining front to the next one's, in a combination of the two.

Note: For a better understanding for the different properties of the Pulse object and their meaning in the scientific sense, the reader is referred to the work of P. Germann.[2]

Display

Animation

As the drawing of the content of the canvas is done in the background and not continuously built up for the user to see, a straight approach to animate each step proved difficult. Therefore, a trick had to be used.

```
<div id="cover-graph"></div>
```

For both *canvases* there is a corresponding div

```
#cover-graph {
	position: absolute;
	width: 800px;
	height: 400px;
	background-color: #fff;
	top: 203px;
	left: 0px;
	z-index: 1;
	background-image: url('raster.gif');
	background-position: top right;
}
```

They are initially is set to be the same *height*, *width*, *position* and *background image* as their *canvas*.

```
function animation(){
	if(!animating){
		animating = true;
		var step = 3;
		var interval = 25;
		var coverGraph = document.getElementById("cover-graph");
		var coverPulses = document.getElementById("cover-pulses");
		var currentWidth = 800;
		var currentLeft = 0;
		var currentTop = 200;
		var currentHeight = 400;
		setTimeout(function(){reduce();}, interval);

		function reduce()
		{
			if (currentWidth > 0 && currentHeight > 0)
			{
				currentWidth -= step;
				currentLeft += step;
				coverGraph.style.width = currentWidth + "px";
				coverGraph.style.left = currentLeft + "px";
				coverPulses.style.width = currentWidth + "px";
				coverPulses.style.left = currentLeft + "px";
				coverGraph.style.display = "block";
				coverPulses.style.display = "block";

				setTimeout(function(){reduce();}, interval);
			}
			else
```

```
                {
                        coverGraph.style.display = "none";
                        coverPulses.style.display = "none";
                        animating = false;
                }
        }
    }
}
```

The covering *div* is then decreased in size by an amount of pixels specified in *step*, repeated each amount of milliseconds specified in *interval*. At the end, the *div* is set to not be visible anymore.

Drawing

The *canvas* is drawn on via the so called *context* of a *canvas*, and can be imagined like a brush.

```
var canvasGraph = document.getElementById("canvas-graph");
[...]
if (null==canvasGraph || !canvasGraph.getContext) return;
var ctx = canvasGraph.getContext("2d");
ctx.beginPath();
ctx.lineWidth = 1;
ctx.strokeStyle = color;
[...]
ctx.moveTo(x1,y1);
ctx.lineTo(x2,y2);
[...]
ctx.stroke();
```

The *context* is retrieved from the *canvas* and a new path is started, and some style information is assigned to the *context*. Then the *context* is moved to a certain point, and a line is generated to another point. When all lines have been generated, the line is drawn or *stroked* on the *canvas*.

```
ctx.fillRect(startx, maxHeight-height, endx-startx, height);
```

If there is a rectangle to be drawn, there is also a function for that, with a start *x* cordinate, a start *y* coordinate, a *width* and a *height*. There is no need to draw the rectangle other than this one function call.

```
ctx.clearRect(0,0,canvasGraph.width, canvasGraph.height);
```

A rectangle is cleared by invoking this function, again with start x, start y, width and height.

Drawing the functions

The *context* can only draw thee shapes; circles, rectangles and straight lines. There are no curved lines or bezier functions. So while the drawing of a linear function would be as easy as drawing a line, the root function's curve couldn't be produced by just that. Therefore the decision was made to draw each function's graph step by step, pixel by pixel.

```
function drawFunction(f, start, end, color){
        if(!color) var color = "#0000ff";
        if (null==canvasGraph || !canvasGraph.getContext) return;
        var ctx = canvasGraph.getContext("2d");
        var xx, yy;
```

```
      var iMax = xMax;
      ctx.beginPath();
      ctx.lineWidth = 2;
      ctx.strokeStyle = color;
 for (var i=start;i<=end;i+=xStep) {
      xx = i*scale.x; yy = scale.y*f(i);
  if (i==start) ctx.moveTo(xx,yy);
  else          ctx.lineTo(xx,yy);
 }
 ctx.stroke();
}
```

Click events

In the area of the *canvas* there are 3 elements which can be clicked on to reveal the legend, the documentation section and the data section of the document, respectively.
These areas all have been assigned an *onclick*-event via Javascript.

```
values = document.getElementById('values');
valuesHidden = true;
values.onclick = function(e) {
	if(valuesHidden){
		values.style.width = "768px";
		values.style.right = "5px";
		values.style.borderRight = "1px solid black";
		valuesHidden = false;
	}else{
		values.style.width = "0px";
		values.style.right = "0px";
		values.style.borderRight = "none";
		valuesHidden = true;
	}
}
```

The elements each have a basic CSS definition based in the linked CSS file. In the event of a click however, they each are assigned additional CSS properties, based on the current status of their visibility.

The buttons are added the functionality to either call the main function with the corresponding pulse array

```
document.getElementById("pulse6").onclick = function(){ main(pulses6) };
```

to change the current value of L and run the current pulse array again

```
document.getElementById("half-l").onclick = function(){ L = L/2;
main(currentPulses) };
```

or to switch the status of the animation.

```
document.getElementById("animation").onclick = function(){
	animate = !animate;
	if(animate){
		this.innerHTML = "Animation ON";
	} else {
		this.innerHTML = "Animation OFF";
	}
};
```

Hover events

Both of the canvas elements have an *onmousemove* and an *onmoseout* event assigned, which define what happens when the mouse pointer hovers over one of them or leaves their area, respectively.

```
document.getElementById("cover-graph").onmousemove = hoverGraph;
document.getElementById("cover-graph").onmouseout = mouseoutGraph;
function hoverGraph(e){
	var div = document.getElementById("hover-graph");
	var that = canvasGraph;
	var x = e.pageX - that.parentElement.offsetLeft - 1;
	var y = e.pageY - that.offsetTop - 0.5;
	if (y>=0 && x >= 0){
		x = x*xStep;
		y = Math.round(y*yStep*1000)/1000;
		div.innerHTML = Math.floor(x/3600) + " h "
			+ Math.floor(x/60%60) + " min "
			+ "<br/>"
			+ y + " m";
		div.style.display = "block";
		div.style.left = (that.offsetLeft + that.offsetWidth - 110) + "px";
		div.style.top = (that.offsetTop + 3) + "px";
		div.style.width = "95px";
		div.style.paddingTop = "20px";
		div.style.paddingBottom = "20px";
	}else {
		div.style.display = "none";
	}
}
function mouseoutGraph(e) {
	var div = document.getElementById("hover-graph");
	div.style.display = "none";
}
```

The elements responsible for displaying the hover-information for each *canvas* possess a basic CSS definition, defined in the CSS file linked in the HTML file.
The event handlers define additional CSS properties, depending on the position of the mouse. Furthermore, they track the position of the mouse, and in dependence manipulate the content of the displaying elements to reflect the changes in position.

The main function

The purpose of the main function is to iterate through every pulse, determine which functions of the current pulse are to be used, where they start and where they stop, and then draw the pulses and their active functions' graph on the *canvas*.

```
for(i=0; i<pulses.length;i++){
      p = pulses[i];
      next = pulses[i+1];
      drawPulse(p);
      for(i=0; i<pulses.length;i++){
[...]
      currentFunction = (function(point, p){return function(t){return
      linear(point, p.v, t)}})(currentPoint, p);
      if(next.intensity > p.intensity){
            nextPoint.x = intersection(currentFunction, function(t){ return
            jump(p,next,t); }, currentPoint.x, xMax);
            nextPoint.y = currentFunction(nextPoint.x);
            drawFunction(function(t){ return jump(p,next,t); }, next.start,
            nextPoint.x, "0ac2ff");
            drawFunction(currentFunction, currentPoint.x, nextPoint.x,
            "#47FF0A");
      }else{
            nextPoint.x = intersection(currentFunction, p.drainagefront,
            currentPoint.x, xMax);
            nextPoint.y = currentFunction(nextPoint.x);
            drawFunction(currentFunction, currentPoint.x, nextPoint.x,
            "#47FF0A");
            drawFunction(p.drainagefront, p.end, nextPoint.x, "0a56ff");
            currentPoint.copy(nextPoint);
            difference = currentFunction(currentPoint.x)-
            p.combined(currentPoint.x);
            currentFunction = (function(p){return function(t){ return
            p.combined(t) + difference }})(p);
            nextPoint.x = intersection(currentFunction, next.drainagefrontStart,
            currentPoint.x, xMax);
            nextPoint.y = currentFunction(nextPoint.x);
            drawFunction(next.drainagefrontStart, next.start, nextPoint.x,
            "0a56ff");
            drawFunction(currentFunction, currentPoint.x, nextPoint.x,
            "#FF0A47");
      }
}
```

This is done for each pulse, with the first and the last one being special cases, the first one starting the graph and the last one not having a successor.

Sources

[1] http://en.wikipedia.org/wiki/Newton's_method
[2] Germann, Peter, unpublished Manuscript.

Appendix A – Full source code

mi.html

```
<!DOCTYPE html>
<!--
/
*******************************************************************************
*****
*     Modeling Infiltration
*     A Bachelor Thesis
*     HTML structure
*
*     author: Ramon Wenger
*     University of Bern, 2012
*     http://scg.unibe.ch
*
*     This file contains the structure of the document the application is
*     built on. It uses HTML5 standard. It is valid HTML5 according to
*     the W3 validator at http://validator.w3.org/check
*******************************************************************************
*****/
 -->
<html>
<head>
<title>Modeling Infiltration</title>
<link href="mi.css" rel="stylesheet" type="text/css">
<meta name="description" content="Illustration of water infiltration in the
soil, made with Javascript and HTML5" />
<meta name="keywords" content="HTML5,Javascript,Water Infiltration" />
<meta name="author" content="Ramon Wenger" />
<meta http-equiv="content-type" content="text/html;charset=UTF-8" />
</head>
<body>
      <div id="wrap">
            <canvas id="canvas-pulses" width="800" height="200"></canvas>
            <canvas id="canvas-graph" width="800" height="400"></canvas>
            <div id="hover-graph"></div>
            <div id="hover-pulse"></div>
            <div id="cover-pulses"></div>
            <div id="cover-graph"></div>
            <div id="values"></div>
            <img id="legende" src="legende.gif" alt="legende" title="legende"/>
            <div id="help">?</div>
            <button id="pulse1">1 Puls</button>
            <button id="pulse2">2 Pulse</button>
            <button id="pulse3">3 Pulse</button>
            <button id="pulse4">4 Pulse</button>
            <button id="pulse6">6 Pulse</button>
            <button id="pulse12">12 Pulse</button>
            <br/>
            <button id="half-l">1/2 * L</button>
            <button id="double-l">2 * L</button>
            <br/>
            <button id="animation">Animation ON</button>
      </div>
```

```
      <script type="text/javascript" src="mi-definitions.js"></script>
      <script type="text/javascript" src="mi-calls.js"></script>
</body>
</html>
```

mi.css

```
/
*******************************************************************************
*****
*     Modeling Infiltration
*     A Bachelor Thesis
*     Layout
*
*     author: Ramon Wenger
*     University of Bern, 2012
*     http://scg.unibe.ch
*
*     This CSS file contains the style and layout information that is
*     applied to the html part of the application
*
*******************************************************************************
*****/
body {
	text-align: center;
	background-color: #333;
	font-family: Verdana,Arial,sans-serif;
	height: 100%;
}
#wrap {
	background-color: #ccc;
	width: 800px;
	margin: 0 auto;
	height: 100%;
	position: relative;
}
canvas {
	background-color: #fff;
}
#canvas-graph {
	background-image: url('raster.gif');
	background-position: top right;
}
#canvas-pulses {
	background-image: url('raster.gif');
	background-position: bottom right;
}
#hover-graph {
	width: 65px;
	display: none;
	position: absolute;
	left: 50px;
	top: 0;
	border: 1px solid black;
	padding-top: 20px;
	padding-right: 10px;
	text-align: left;
	font-size: 0.75em;
	padding-bottom: 20px;
	z-index: 5;
	text-align: right;
	background-color: #ffffff;
}
```

```
#hover-pulse {
	width: 100px;
	display: none;
	position: absolute;
	left: 50px;
	top: 0;
	border: 1px solid black;
	padding-top: 20px;
	padding-left: 10px;
	text-align: left;
	font-size: 0.75em;
	padding-bottom: 20px;
	background-color: #ffffff;
	z-index: 5;
}
#cover-graph {
	position: absolute;
	width: 800px;
	height: 400px;
	background-color: #fff;
	top: 203px;
	left: 0px;
	z-index: 1;
	background-image: url('raster.gif');
	background-position: top right;
}
#cover-pulses {
	position: absolute;
	width: 800px;
	height: 200px;
	background-color: #fff;
	top: 0px;
	left: 0px;
	z-index: 1;
	background-image: url('raster.gif');
	background-position: bottom right;
}
#values {
	position: absolute;
	width: 0px;
	height: 580px;
	background-color: #fff;
	top: 0px;
	right: 0px;
	z-index: 2;
	overflow: hidden;
	padding-left: 20px;
	padding-top: 10px;
	border: 1px solid black;
	border-right: none;
	margin-top: 5px;
	cursor: e-resize;
	text-align: left;
	z-index: 4;
	font-size: 0.8em;
}
#values table {
	text-align: right;
}
#values table td, #values table th , #values span{
	width: 150px;
}
#values span {
	display: block;
```

```
        float: left;
        margin-left: 2px;
        padding: 1px;
        text-align: right;
}
#values br {
        clear: both;
}
#legende {
        position: absolute;
        left: 5px;
        top: 500px;
        width: 100px;
        opacity: 0.4;
        z-index: 3;
        cursor: ne-resize;
}
#values span.super {
        font-size: 0.83em;
        vertical-align: super;
        width: auto;
}
#help {
        position: absolute;
        top: 5px;
        left: 5px;
        background-color: #fff;
        font-weight: bold;
        border: 1px solid #000;
        padding: 2px;
        cursor: nw-resize;
        z-index: 5;
}
```

mi-definitions.js

```
/
**********************************************************************************
*****
*     Modeling Infiltration
*     A Bachelor Thesis
*     Definitions
*
*     author: Ramon Wenger
*     University of Bern, 2012
*     http://scg.unibe.ch
*
*     This file contains the definitions of the functions and variables
*     of the application
*
**********************************************************************************
*****/

/
**********************************************************************************
*****
*
*     GLOBAL VARIABLES AND HTML ELEMENTS
*
**********************************************************************************
*****/
var L = 1130;
var xMax = 90000*3;
var maxIntensity = 0.0000022;
var accuracy = 1000;
var K = 3.27 * Math.pow(10,6); //Konstante, für g/3*n (g = Gravitation, n =
Viskosität)
var xStep = xMax/800;
var yStep = 0.1;
var pulses = [];
var   scale = {
      x : 1/xStep,
      y : 1/yStep,
};
currentPulses = [];
var canvasPulses = document.getElementById("canvas-pulses");
var canvasGraph = document.getElementById("canvas-graph");
/
**********************************************************************************
*****
*
*     OBJECTS
*
**********************************************************************************
*****/
function Pulse(start, end, intensity){
      this.start = start;
      this.end = end;
      this.intensity = intensity; // q
      var F, w, v, c;

      var pulse = this;
      this.schnittpunkt = new Point(0,0);
```

```
	this.jump = function(t) { return null; };


	this.feuchtefront = function(t) {
		return pulse.v*(t-pulse.start);
	}

	this.setVars = function(){
		pulse.F = Math.pow(pulse.intensity/(K*L),1/3);
		pulse.w = L*pulse.F;
		pulse.v = K*Math.pow(pulse.F,2);
		pulse.c = 3*pulse.v;
	}

	this.drainagefrontStart = function(t){
		return pulse.c*(t-pulse.start);
	}

	this.drainagefront = function(t) {
		return pulse.c*(t-pulse.end);
	}

	function setSchnittpunkt(){
		pulse.schnittpunkt.x = intersection(pulse.feuchtefront,
pulse.drainagefront, pulse.start, xMax);
		pulse.schnittpunkt.y = pulse.feuchtefront(pulse.schnittpunkt.x);
	}

	this.combined = function(t){
		//todo: nur für t>pulse.end?
		return pulse.c*Math.pow((pulse.end-pulse.start)/2,2/3)*Math.pow(t-
pulse.end,1/3);
	}

	this.setEnd = function(newEnd){
		pulse.end = newEnd;
		setSchnittpunkt();
	}

	this.setJump = function(p){
		pulse.jump = function(t){
			cj = K*(Math.pow(p.F,2)+p.F*pulse.F+Math.pow(pulse.F,2));
			return cj*(t-pulse.end);
		}
	}

	this.setIntensity = function(i){
		pulse.intensity = i;
		pulse.setVars();
	}

	this.setVars();
	setSchnittpunkt();
}
function Point(x, y){
	this.x = x;
	this.y = y;

	this.copy = function(point){
		this.x = point.x;
		this.y = point.y;
	}
}
/
```

```
*****************************************************************************
*****
*
*     APPLICATION LOGIC
*
*****************************************************************************
*****/
Array.prototype.top = function(){return this[this.length-1]};
function intersection(f1, f2, min, max){
      if(Math.floor(f1(min)*accuracy) == Math.floor(f2(min)*accuracy)){
            return min;
      }else if(Math.floor(f1(max)*accuracy) == Math.floor(f2(max)*accuracy)){
            return max;
      }else if( (f1(min)>f2(min) && f1(max)>f2(max)) || (f1(min)<f2(min) &&
f1(max)<f2(max)) ){
            return false;
      }else{
            if(intersection(f1,f2,min+(max-min)/2,max)){
                  return intersection(f1,f2,min+(max-min)/2,max);
            }else{
                  return intersection(f1,f2,min,max-(max-min)/2);
            }
      }
}
function jump(p1, p2, t){
      c = K*(Math.pow(p1.F,2)+p1.F*p2.F+Math.pow(p2.F,2));
      return c*(t-p2.start);
}
function linear(start, slope, t){
      return start.y+slope*(t-start.x);
}

function getMaxIntensity(pulses){
      max = 0;
      for(i=0;i<pulses.length;i++){
            if(pulses[i].intensity>max)
                  max = pulses[i].intensity;
      }
      return max;
}
function setVars(pulses){
      for(i=0;i<pulses.length;i++){
            pulses[i].setVars();
      }
      return pulses;
}
/
*****************************************************************************
*****
*
*     CANVAS EVENT HANDLERS
*
*****************************************************************************
*****/
canvasGraph.onmousemove = hoverGraph;
canvasGraph.onmouseout = mouseoutGraph;
canvasPulses.onmousemove = hoverPulse;
canvasPulses.onmouseout = mouseoutPulse;
document.getElementById("legende").onmousemove = hoverGraph;
document.getElementById("cover-graph").onmousemove = hoverGraph;
document.getElementById("cover-graph").onmouseout = mouseoutGraph;
function hoverGraph(e){
      var div = document.getElementById("hover-graph");
      var that = canvasGraph;
```

```
	var x = e.pageX - that.parentElement.offsetLeft - 1;
	var y = e.pageY - that.offsetTop - 0.5;
	if (y>=0 && x >= 0){
		x = x*xStep;
		y = Math.round(y*yStep*1000)/1000;
		div.innerHTML = Math.floor(x/3600) + " h "
			+ Math.floor(x/60%60) + " min "
			+ "<br/>"
			+ y + " m";
		div.style.display = "block";
		div.style.left = (that.offsetLeft + that.offsetWidth - 110) + "px";
		div.style.top = (that.offsetTop + 3) + "px";
		div.style.width = "95px";
		div.style.paddingTop = "20px";
		div.style.paddingBottom = "20px";
	}else {
		div.style.display = "none";
	}
}
function mouseoutGraph(e) {
	var div = document.getElementById("hover-graph");
	div.style.display = "none";
}
function hoverPulse(e) {
	var div = document.getElementById("hover-pulse");
	var x = e.pageX - this.parentElement.offsetLeft - 1;
	var y = e.pageY - this.parentElement.offsetTop - 0.5;
	if(pulse = checkPulse(x,y)){
		div.innerHTML = "start: " + Math.floor(pulse.start/3600) + " h " +
Math.floor(pulse.start/60 % 60) + " min" +
					"<br/>end: " + Math.floor(pulse.end/3600) +
" h " + Math.floor(pulse.end/60 % 60) + " min" +
					"<br/>intensity: " + pulse.intensity + "
m/s" +
					"<br/>F: " +
Math.round(pulse.F*10000000)/10000000 +
					"<br/>v: "
+Math.round(pulse.v*10000000)/10000000 +
					"<br/>c: " +
Math.round(pulse.c*10000000)/10000000;
		div.style.display = "block";
		div.style.width = "150px";
		div.style.paddingTop = "5px";
		div.style.paddingBottom = "5px";
		div.style.left = (this.offsetLeft + this.offsetWidth -
div.offsetWidth - 3) + "px";
		div.style.top = (this.offsetTop + this.offsetHeight -
div.offsetHeight - 3) + "px";
	} else {
		div.style.display = "none";
	}
}
function mouseoutPulse(e) {
	var div = document.getElementById("hover-pulse");
	div.style.display = "none";
}
values = document.getElementById('values');
valuesHidden = true;
values.onclick = function(e) {
	if(valuesHidden){
		values.style.width = "768px";
		values.style.right = "5px";
		values.style.borderRight = "1px solid black";
		valuesHidden = false;
```

```
	}else{
		values.style.width = "0px";
		values.style.right = "0px";
		values.style.borderRight = "none";
		valuesHidden = true;
	}
}
legendeSmall = true;
document.getElementById("legende").onclick = function(e){
	if(legendeSmall){
		this.style.width = "350px";
		this.style.top = "300px";
		this.style.opacity = "1";
		legendeSmall = false;
	}else{
		this.style.width = "100px";
		this.style.opacity = "0.4";
		this.style.top = "500px";
		legendeSmall = true;
	}
}
helpSmall = true
document.getElementById("help").onclick = function(e){
	if(helpSmall){
		this.style.width = "785px";
		this.style.height = "588px";
		this.style.fontWeight = "normal";
		this.innerHTML = "<br/><br/>Documentation goes here";
		helpSmall = false;
	}else{
		this.style.width = "auto";
		this.style.height = "auto";
		this.style.fontWeight = "bold";
		this.innerHTML = "?";
		helpSmall = true;
	}
}
/
**************************************************************************************
*****
*
*	DISPLAY FUNCTIONS
*
**************************************************************************************
*****/
function animation(){
	if(!animating){
		animating = true;
		var step = 3;
		var interval = 25;
		var coverGraph = document.getElementById("cover-graph");
		var coverPulses = document.getElementById("cover-pulses");
		var currentWidth = 800;
		var currentLeft = 0;
		var currentTop = 200;
		var currentHeight = 400;
		setTimeout(function(){reduce();}, interval);

		function reduce()
		{
			if (currentWidth > 0 && currentHeight > 0)
			{
				currentWidth -= step;
				currentLeft += step;
```

```
                        coverGraph.style.width = currentWidth + "px";
                        coverGraph.style.left = currentLeft + "px";
                        coverPulses.style.width = currentWidth + "px";
                        coverPulses.style.left = currentLeft + "px";
                        coverGraph.style.display = "block";
                        coverPulses.style.display = "block";

                        setTimeout(function(){reduce();}, interval);
                  }
                  else
                  {
                        coverGraph.style.display = "none";
                        coverPulses.style.display = "none";
                        animating = false;
                  }
            }
      }
}
function clearCanvas(){
      var  ctxPulses=canvasPulses.getContext("2d");
      var  ctxGraph=canvasGraph.getContext("2d");

      ctxPulses.clearRect(0,0,canvasPulses.width, canvasPulses.height);
      ctxGraph.clearRect(0,0,canvasGraph.width, canvasGraph.height);
}
function drawRec(startx, maxHeight, endx, height) {
      if (null==canvasPulses || !canvasPulses.getContext) return;

      var  ctx=canvasPulses.getContext("2d");
      ctx.fillStyle = "#0000ff";
      ctx.fillRect(startx, maxHeight-height, endx-startx, height);
      return false;
 }
function drawPulse(pulse){
      if(pulse.intensity>0){
            drawRec(pulse.start/xStep, canvasPulses.height, pulse.end/xStep,
pulse.intensity*canvasPulses.height/maxIntensity);
      }
}
function drawFunction(f, start, end, color){
      if(!color) var color = "#0000ff";
      if (null==canvasGraph || !canvasGraph.getContext) return;
      var ctx = canvasGraph.getContext("2d");
      var xx, yy;
      var iMax = xMax;
      ctx.beginPath();
      ctx.lineWidth = 2;
      ctx.strokeStyle = color;
 for (var i=start;i<=end;i+=xStep) {
      xx = i*scale.x; yy = scale.y*f(i);
  if (i==start) ctx.moveTo(xx,yy);
  else          ctx.lineTo(xx,yy);
 }
 ctx.stroke();
}
function checkPulse(x,y){
      var pulse;
      for (i = 0; i< currentPulses.length;i++){
            pulse = currentPulses[i];
                  if((x>pulse.start/xStep && x<pulse.end/xStep) &&
(y<canvasPulses.height && y>canvasPulses.height-
pulse.intensity*canvasPulses.height/maxIntensity) )
                   return pulse;
      }
```

```
        return 0;
}
```

mi-calls.js

```
/
*****************************************************************************
*****
*     Modeling Infiltration
*     a Bachelor Thesis
*     Calls
*
*     author: Ramon Wenger
*     University of Bern, 2012
*     http://scg.unibe.ch
*
*     This file contains the creation of the pulses and the call of the
*     main() function
*****************************************************************************
*****/

/
*****************************************************************************
*****
*
*     PULSE DEFINITIONS
*
*****************************************************************************
*****/
var pulses12 = [];
pulses12.push(new Pulse(0,2800,0));
pulses12.push(new Pulse(2800,9200,0.00000017));
pulses12.push(new Pulse(9200,12800,0.000001));
pulses12.push(new Pulse(12800,19400,0.0000008));
pulses12.push(new Pulse(19400,29800,0.0000014));
pulses12.push(new Pulse(29800,39600,0.0000008));
pulses12.push(new Pulse(39600,45400,0.0000014));
pulses12.push(new Pulse(45400,50000,0.0000022));
pulses12.push(new Pulse(50000,58000,0.0000008));
pulses12.push(new Pulse(58000,65000,0.0000006));
pulses12.push(new Pulse(65000,75000,0.0000004));
pulses12.push(new Pulse(75000,83000,0.0000003));
pulses12.push(new Pulse(83000,90000,0.0000001));

pulses1 = [];
pulses1.push(new Pulse(0, 2800, 0));
pulses1.push(new Pulse(2800, 90000, 8.22e-07));

pulses2 = [];
pulses2.push(new Pulse(0, 2800, 0));
pulses2.push(new Pulse(2800, 45400, 9.50e-07));
pulses2.push(new Pulse(45400, 90000, 7.51e-07));

pulses3 = [];
pulses3.push(new Pulse(0, 2800, 0));
pulses3.push(new Pulse(2800, 29800, 9.08e-07));
pulses3.push(new Pulse(29800, 58000, 1.15e-06));
pulses3.push(new Pulse(58000, 90000, 5.31e-07));

pulses4 = [];
pulses4.push(new Pulse(0, 2800, 0));
pulses4.push(new Pulse(2800, 19400, 6.00e-07));
```

```
pulses4.push(new Pulse(19400, 45400, 1.17e-06));
pulses4.push(new Pulse(45400, 65000, 9.50e-07));
pulses4.push(new Pulse(65000, 90000, 5.96e-07));

pulses6 = [];
pulses6.push(new Pulse(0, 2800, 0));
pulses6.push(new Pulse(2800, 12800, 4.68e-07));
pulses6.push(new Pulse(12800, 29800, 1.16e-06));
pulses6.push(new Pulse(29800, 45400, 1.02e-06));
pulses6.push(new Pulse(45400, 58000, 1.31e-06));
pulses6.push(new Pulse(58000, 75000, 4.17e-07));
pulses6.push(new Pulse(75000, 90000, 6.59e-07));

animate = true;
/
*************************************************************************
*****
*
*     BUTTON EVENT HANDLERS
*
*************************************************************************
*****/
document.getElementById("pulse1").onclick = function(){ main(pulses1) };
document.getElementById("pulse2").onclick = function(){ main(pulses2) };
document.getElementById("pulse3").onclick = function(){ main(pulses3) };
document.getElementById("pulse4").onclick = function(){ main(pulses4) };
document.getElementById("pulse6").onclick = function(){ main(pulses6) };
document.getElementById("pulse12").onclick = function(){ main(pulses12) };
document.getElementById("half-l").onclick = function(){ L = L/2;
main(currentPulses) };
document.getElementById("double-l").onclick = function(){ L = L*2;
main(currentPulses) };
document.getElementById("animation").onclick = function(){
      animate = !animate;
      if(animate){
            this.innerHTML = "Animation ON";
      } else {
            this.innerHTML = "Animation OFF";
      }
};
animating = false;
main(pulses2);
animation();
function main(pulses){
      currentPulses = pulses;
      pulses = setVars(pulses);
      clearCanvas();
      difference = 0;
      points = [];
      points.push(new Point(0,0));
      currentFunction = function(){return 0;};
      inter = 0;
      functions = [];
      nextPoint = new Point(0,0);

      for(i=0; i<pulses.length;i++){
            p = pulses[i];
            next = pulses[i+1];
            drawPulse(p);
            if(i==1 && pulses.length - 1 != 1){
                  if(next.intensity > p.intensity){
                        nextPoint.x = intersection(p.feuchtefront, function(t){
return jump(p,next,t); }, p.start, xMax);
                        nextPoint.y = p.feuchtefront(nextPoint.x);
```

```
                        points.push(new Point(nextPoint.x,nextPoint.y));
                        drawFunction(function(t){ return jump(p,next,t); },
next.start, nextPoint.x, "0AC2FF");
                        drawFunction(p.feuchtefront, p.start, nextPoint.x,
"#47FF0A");
                        (function(p){ functions.push(p.feuchtefront) })(p);
                  }else{
                        nextPoint.x = p.schnittpunkt.x;
                        nextPoint.y = p.schnittpunkt.y;
                        points.push(p.schnittpunkt);
                        drawFunction(p.drainagefront, next.start, nextPoint.x,
"0a56ff");
                        drawFunction(p.feuchtefront, p.start, nextPoint.x,
"#47FF0A");
                        (function(p){ functions.push(p.feuchtefront)})(p);
                        currentPoint = points.top();
                        currentFunction = p.combined;
                        (function(p){ functions.push(p.combined)})(p);
                        nextPoint.x = intersection(currentFunction,
next.drainagefrontStart, p.start, xMax);
                        nextPoint.y = currentFunction(nextPoint.x);
                        points.push(new Point(nextPoint.x, nextPoint.y));
                        drawFunction(currentFunction, currentPoint.x,
nextPoint.x, "#FF0A47");
                        drawFunction(next.drainagefrontStart, next.start,
nextPoint.x, "#0a56ff");
                  }
            } else if(i>=2 && i < pulses.length - 1){
                        currentPoint = points.top();
                        currentFunction = (function(point, p){return function(t)
{return linear(point, p.v, t)}})(currentPoint, p);
                  if(next.intensity > p.intensity){
                        nextPoint.x = intersection(currentFunction, function(t){
return jump(p,next,t); }, currentPoint.x, xMax);
                        nextPoint.y = currentFunction(nextPoint.x);
                        drawFunction(function(t){ return jump(p,next,t); },
next.start, nextPoint.x, "0ac2ff");
                        drawFunction(currentFunction, currentPoint.x,
nextPoint.x, "#47FF0A");
                        points.push(new Point(nextPoint.x, nextPoint.y));
                        functions.push(currentFunction);
                  }else{
                        nextPoint.x = intersection(currentFunction,
p.drainagefront, currentPoint.x, xMax);
                        nextPoint.y = currentFunction(nextPoint.x);
                        drawFunction(currentFunction, currentPoint.x,
nextPoint.x, "#47FF0A");
                        drawFunction(p.drainagefront, p.end, nextPoint.x,
"0a56ff");
                        points.push(new Point(nextPoint.x, nextPoint.y));
                        functions.push(currentFunction);
                        currentPoint.copy(nextPoint);
                        difference = currentFunction(currentPoint.x)-
p.combined(currentPoint.x);
                        currentFunction = (function(p){return function(t){
return p.combined(t) + difference }})(p);
                        nextPoint.x = intersection(currentFunction,
next.drainagefrontStart, currentPoint.x, xMax);
                        nextPoint.y = currentFunction(nextPoint.x);
                        drawFunction(next.drainagefrontStart, next.start,
nextPoint.x, "0a56ff");
                        points.push(new Point(nextPoint.x, nextPoint.y));
                        drawFunction(currentFunction, currentPoint.x,
nextPoint.x, "#FF0A47");
```

```
                        functions.push(currentFunction);
                    }
                } else if(i==pulses.length-1){
                        currentPoint = points.top();
                        currentFunction = (function(point, p){return function(t)
{return linear(point, p.v, t)}})(currentPoint, p);
                        nextPoint.x = intersection(currentFunction, p.drainagefront,
currentPoint.x, xMax);
                        nextPoint.y = currentFunction(nextPoint.x);
                        drawFunction(currentFunction, currentPoint.x, nextPoint.x,
"#47FF0A");
                        drawFunction(p.drainagefront, p.end, nextPoint.x, "0a56ff");
                        points.push(nextPoint.x, nextPoint.y);
                        functions.push(currentFunction);
                        currentPoint.copy(nextPoint);
                        difference = currentFunction(currentPoint.x)-
p.combined(currentPoint.x);
                        currentFunction = (function(p){return function(t){ return
p.combined(t) + difference }})(p);
                        nextPoint.x = xMax;
                        nextPoint.y = currentFunction(nextPoint.x);
                        points.push(new Point(nextPoint.x, nextPoint.y));
                        drawFunction(currentFunction, currentPoint.x, nextPoint.x,
"#FF0A47");
                        functions.push(currentFunction);
                }
        }
        html =
"<table><tr><th>Pulse</th><th>start</th><th>end</th><th>intensity</th></tr>";
        for(i=0;i<pulses.length;i++){
                p=pulses[i];
                html += "<tr><td>"+i+"</td><td>"+Math.floor(p.start/3600) +" h " +
Math.floor(p.start/60 % 60) +
                " min</td><td>"+Math.floor(p.end/3600) +" h " + Math.floor(p.end/60
% 60) +
                " min</td><td>" + p.intensity + " m/s</td></tr>";
        }
        html += "</table>";
        html += "<br/><span>maximal depth</span><span>"+
Math.floor(functions.top()(xMax)*100)/100 +" m</span>";
        html += "<br/><span>timeframe</span><span>"+ xMax/3600 + "h " + xMax/60 %
60 + " min</span>";
        html += "<br/><span>value of L</span><span>"+ L +" m</span><span
class=\"super\">-1</span>";
        values.innerHTML = html;
        if(animate)
                animation();
}
```