

# **FSPM-Prototype V0.02 for GroIMP User Manual**

**Michael Henke and Gerhard H. Buck-Sorlin**

June 27, 2010

# FSPM-Prototype V0.02 for GroIMP

## User Manual

---

Michael Henke<sup>1</sup> and Gerhard Buck-Sorlin<sup>2</sup>

<sup>1</sup> Dept. Ecoinformatics, Biometrics & Forest Growth, Büsgen Institute  
Georg-August-University Göttingen, Büsgenweg 4, 37077 Göttingen, Germany  
email: mhenke@uni-goettingen.de

<sup>2</sup> Wageningen UR, Department Biometris, Droevendaalsesteeg 1  
6708 PB Wageningen, The Netherlands  
email: gerhard.buck-sorlin@wur.nl

Version: V0.02

Date: June 27, 2010 – 0:59

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.2	Existing and envisaged models based on FSPM-Prototype . . . . .	4
<b>2</b>	<b>Model description</b>	<b>5</b>
2.1	The FSPM-Prototype . . . . .	5
2.1.1	Features . . . . .	6
2.1.2	General processes . . . . .	6
	Source activity . . . . .	6
	Sink activity . . . . .	7
2.1.3	Vegetative and generative development . . . . .	8
2.1.4	Dynamics of the central assimilate pool . . . . .	9
2.1.5	Radiation model and light interception . . . . .	10
2.2	File description . . . . .	10
2.2.1	Model . . . . .	11
	Parameter.rgg . . . . .	11
	Parameters: . . . . .	12
	Environment parameters: . . . . .	12
	Methods: . . . . .	13
	main.rgg . . . . .	13
	Parameters: . . . . .	14
	Methods: . . . . .	14
	Additional Methods: . . . . .	15
	OrgansS1.rgg . . . . .	15
	Parameter (constants): . . . . .	15
	Methods: . . . . .	16
	Organ - Interface . . . . .	16
	Organ - Parameters . . . . .	17
	Individual - Module . . . . .	17
	ShootO - Module . . . . .	19
	Seed . . . . .	19
	Root . . . . .	20
	Bud . . . . .	20
	Internode . . . . .	21
	Leaf . . . . .	22
	Peduncle . . . . .	24
	Flower . . . . .	25
	Fruit . . . . .	25

RulesS1.rgg . . . . .	26
Methods: . . . . .	26
SpeciesParameterS1.rgg . . . . .	28
Parameter (constants): . . . . .	28
2.2.2 Photosynthesis models . . . . .	29
Files: . . . . .	29
Description: . . . . .	29
General . . . . .	29
LEAFC3 / -N . . . . .	30
Kim and Lieth . . . . .	31
Lieth and Pasian . . . . .	31
Thornley . . . . .	31
2.2.3 Other . . . . .	31
Charts.rgg . . . . .	31
Tools.rgg . . . . .	32
Modules: . . . . .	32
Mathematical functions: . . . . .	32
Additional functions: . . . . .	35
Readme.txt . . . . .	35
MyFileChooser.java . . . . .	35
ToDo.txt . . . . .	35
Changes.log . . . . .	35
<b>3 Building A New Model</b>	<b>37</b>
3.1 Single plant model . . . . .	38
3.2 Multiple species . . . . .	38
3.3 Experiments to try . . . . .	39
3.3.1 Environment conditions . . . . .	39
3.3.2 Model conditions . . . . .	40
<b>4 Experimental Settings</b>	<b>41</b>
4.1 General . . . . .	41
4.2 Single Plant Model . . . . .	42
4.3 Temperature Sensitive Model . . . . .	42
<b>5 Measurement protocol</b>	<b>45</b>
5.1 Equipment . . . . .	45
5.2 Vegetative stages . . . . .	46
5.2.1 Destructive measurements in the field and climate chamber: . . . . .	46
5.2.2 Non-destructive measurements including scanning: . . . . .	47
5.2.3 Photosynthesis measurements . . . . .	48
5.2.4 Measurement of microclimate . . . . .	48
5.3 Generative stages . . . . .	48
5.3.1 Destructive measurements: . . . . .	48
<b>6 Time schedule</b>	<b>51</b>
<b>7 2D Digital Measurements</b>	<b>53</b>

7.1	Image Capture	53
7.1.1	Equipment	54
7.1.2	Measurement Protocol	55
7.2	Leaf Image Analysis	56
7.2.1	Surface area	56
7.2.2	Other measurements	58
7.2.3	Semiautomatic Measurements	58
<b>8</b>	<b>Data processing</b>	<b>61</b>
8.1	Data processing and storage	61
8.2	Model specific	62
8.2.1	Conversion of dry weight to fresh weight	62
8.2.2	Average Dry weight	62
8.2.3	Seed fertilization rate	62
8.2.4	Microclimate / Environment	63
<b>9</b>	<b>Results</b>	<b>65</b>
9.1	Morphology	65
9.2	Charts	65
<b>A</b>	<b>Parameter List</b>	<b>69</b>
A.1	OrgansS1.rgg	69
A.1.1	Organ Parameters:	69
A.1.2	Individual module:	69
A.1.3	Root module:	70
A.1.4	Bud module:	70
A.1.5	Internode module:	70
A.1.6	Leaf module:	70
A.1.7	Peduncle module:	71
A.1.8	Flower module:	71
A.1.9	Fruit module:	71
A.2	RulesS1.rgg	71
A.2.1	Plant Parameters:	71
A.3	Parameter.rgg	71
A.4	main.rgg	72
A.5	Charts.rgg	72
A.6	Others	72
A.6.1	Textures	72
<b>B</b>	<b>Templates</b>	<b>73</b>
B.1	Time Schedule (Project)	74
B.2	Time Schedule (Measurements)	75
B.2.1	Vegetative stages	75
B.2.2	Generative stages	76
B.3	Daily measurements:	78
B.4	Destructive measurements (Vegetative):	79
B.5	Non-destructive measurements (Vegetative):	82
B.6	Photosynthesis measurements	85

<b>C</b>	<b>Weather data</b>	<b>87</b>
<b>D</b>	<b>Textures</b>	<b>89</b>
<b>E</b>	<b>Bugs and Problems</b>	<b>91</b>
<b>F</b>	<b>Version History</b>	<b>93</b>
<b>G</b>	<b>Symbols and units</b>	<b>95</b>
<b>H</b>	<b>Glossary</b>	<b>97</b>
H.1	Plant specific . . . . .	97
H.2	Data types . . . . .	98
H.3	Mathematical . . . . .	98
H.4	Growth model . . . . .	99

## List of Figures

2.1	Reallocation of assimilates for growth as a function of relative sink strength of an organ, i. e. relative potential growth rate of that organ, divided by the sum of the potential growth rates of all competing organs . . . . .	9
2.2	Dynamics of carbon . . . . .	10
2.3	Model overview: Files separated by functionality . . . . .	11
2.4	Maximal weight per rank (used function: $a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3$ ) . . . .	22
2.5	Maximal weight per rank (used function: $a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3$ ) . . . .	23
7.1	Examples of some problematic images . . . . .	54
8.1	Example of a directory structure for data storage . . . . .	62
9.1	Seedling, X degree-days old. . . . .	65
9.2	Mature flowering plant (age: X degree-days). . . . .	66
9.3	Final stage of the FSPM-P after X degree-days. . . . .	66
9.4	Dynamics of plant organ numbers during the simulated growth period. . . . .	66
9.5	Potential vs. actual growth rate of leaves. . . . .	67
9.6	Dynamics of leaf length. . . . .	67
9.7	Growth and maintenance respiration ( $g_C$ ). . . . .	67
C.1	Weather data form Meteostation Haarweg - Wageningen 2008 . . . . .	88
a	Temperature . . . . .	88
b	Temperature sum . . . . .	88
c	Sun shine . . . . .	88
d	Radiation . . . . .	88
e	Wind speed . . . . .	88
f	Humidity . . . . .	88

D.1 Juvenile Leaf . . . . . 89

D.2 Adult Leaf . . . . . 89

D.3 Texture of an internode. This is also used for the peduncles. . . . . 90





# Preamble

The present model and the corresponding user manual are still at an early stage of development and thus incomplete.

In order to develop it further, we need your assistance. Therefore, if you find anything that is:

- unclear,
- wrong, or unnecessary,
- missing,
- hard to find,
- inconsistent,
- 
- ...

please feel free to send us an e-mail with your feedback and we will try to fix or include it in the next version.

The Developer Team



# Chapter 1

## Introduction

### 1.1 Introduction

Current crop growth models are often based on a selection of general processes describing the mechanisms of primary production. Generally, in these models factors determining potential, attainable and actual crop growth are distinguished, allowing the use of these models for a variety of crop species, given the availability of a standard set of crop parameters. In contrast to these traditional models, Functional-Structural Models of crops have been developed in a much more ad hoc way, i.e. unsystematically, using a combination of structure-determining rules and at best an arbitrary selection of physiological functions. However, there are no obvious reasons for this: Most physiological functions used in crop models could be used in the same general way in FSPM, and structures, such as plant organs, can be defined generally and then implemented for a crop species.

In botanical research as well as in agricultural and horticultural practice (growers, breeders, consultants, ..), there is an increasing demand for FSPM for diverse purposes. In order to satisfy this demand new, more economic ways of generating FSPMs are needed (TODO: examples). A modular prototype FSPM containing all necessary elements and a basic set of structure-generating rules as well as modules to describe primary production and growth, enormously alleviates the work load involved during the development of an FSPM. Such a prototype model (as will be shown in the following) includes predefined (reusable) parts and thus can be used to quickly arrive at a new model. Furthermore, due to its modular design, it rationalises the work flow and helps to design suitable experiments necessary for obtaining the right set of correlated model parameters.

Building a Functional Structural Plant Model (FSPM) can be a real challenge: Once a suitable set of structural rules is established, there will be additional challenges when it comes to properly parameterizing and calibrating the model. This general FSPM - Prototype (FSPM-P) and the corresponding user manual represent our attempt to provide a structured prototype of a generalized crop plant FSPM, which is easy to reparameterize and expandable. It is based on our experiences with models of crops such as barley (Buck-Sorlin et al. [4], gbs2007, pma06,

barley, Smoleňová, Buck-Sorlin [?]), rose (Buck-Sorlin [2]), tomato (Buck-Sorlin [?]), rice (Xu et al. [26, 25]), poplar (Buck-Sorlin et al. [6])) and several related models (beech (Hemmerling et al. [10]), spruce (Kniemeyer [?]), oilseed rape (Groer [8], Groer et al. [9])).

Here we present a general FSPM written in the language XL and implemented on the software platform GroIMP. This prototype is a clearly structured, coordinated, consistent "all-in-one" system, subdivided into two main parts: a documentation including measurement protocols, time schedules, sections about digital measurements and data processing as well as some templates and the object orientated FSPM-Prototype with its predefined plant organs and integrated functions proper.

A general FSPM can be an intuitive and versatile tool, usable for prototyping, teaching, as a research tool in production systems such as intercropping, as a decision support system, or to enhance our understanding of physiological processes taking place at different hierarchical levels (e.g. organ – individual – canopy). In addition, such an approach can help to establish a standard to make models more transparent and comparable, also for other researchers in the FSPM community.

## 1.2 Existing and envisaged models based on FSPM-Prototype

1. Cotton model: Gerhard Buck-Sorlin, Lizhen Zhang; 2010/11, Beijing, China
2. Rapeseed model: Tian Tian, Wu Lingtong, Gerhard Buck-Sorlin, Michael Henke; 2010/11, Hangzhou, China

# Chapter 2

## Model description

### 2.1 The FSPM-Prototype

Functional Structural Plant Modelling - Prototype (FSPM-P) V0.02 is a framework for FSPM of crop plants which can be easily parametrized. It has been designed and tested under GroIMP 1.1. [17, 14, 16, 13, 15, 18].

The FSPM-P is a working FSPM of a generalized crop plant and provides all necessary elements to do:

- Modelling of individual plants,
- Modelling plant stands (canopies),
- Modelling of selected temperature-sensitive processes,
- Modelling variations in morphology under certain growth conditions.

The file you have downloaded from our site includes three separate files: this manual, the GroIMP model and an example of a climate file (*climate.txt*). In the following section the GroIMP model will be explained in detail.

The model is calibrated for a time period of one year and includes a crop with a growing period of a few months at a selectable resolution of one day or one hour. However, an extension beyond one year is unconditionally possible. This model reproduces plant architecture and morphology of our crop from the seedling stage to harvest maturity.

Although the present model is still at an early stage with respect to the complexity of physiological processes considered, it already contains the main elements. In the following sections the features of the current model are described.

The current version of this generalized crop plant model comes with a set of generalized parameters which need to be replaced by the user with his/her own data. Refer chapter 3, Building

A New Model, for details.

### 2.1.1 Features

The FSPM-P is a fairly extensive set of XL modules and Java implementations comprising:

- elements for a general setup of the system (initiation of the plant individuals, loading parameters, initiation of output charts),
- a module file listing predefined plant organs (object oriented – all organs implement the same interface and contain methods associated with objects),
- vegetative (leaf and internode development) and ...
- ...generative morphology (flower development and fruit formation),
- aggregating objects like shoot module and individual plant module containing summary functions based on XL-queries (e.g. for whole-plant photosynthesis),
- photosynthesis (library of photosynthesis rate models),
- light interception (based on a Monte-Carlo radiation model described in [10]),
- parameters (mainly environmental),
- selectable hourly or daily resolution,
- selectable manual or time controlled harvest function,
- temperature sensitive, different temperature classes,
- statistical output (about 15 predefined charts and file output),
- statistical output to compare individuals

Additional features:

- Each organ type is associated with a different layer thereby allowing to hide certain organ types by deselecting layers in the 3D-view. (By default all layers are selected).
- 

### 2.1.2 General processes

Growth and development are based on source (leaf photosynthesis of assimilates and local storage in a central pool) and sink functions (reallocation of assimilates for growth as a function of sink strength, i. e. relative potential growth rate).

Photosynthesis in the model is restricted to leaf blades, potential photosynthesis of sheaths, stems and walls of immature fruits is not considered, would, however, be possible without problem as all these organs implement the organ interface.

#### Source activity

The main and largely exclusive carbon source for a plant are the leaves (after the carbon stored in the seed has been consumed during germination).

Integrated into the model is a library of photosynthesis rate models (differing in complexity from simple light-response curves to biochemical Farquhar-type models), which can be exchanged / selected by a global parameter, see section 2.2.2.

### Sink activity

The timing and growth duration of active sinks drives the conversion of assimilates to harvestable dry matter. In our FSPM approach the orchestration of sink activity is prescribed by growth and development rules, and the overall biomass production is an emergent property of the integration of these rules applied over time to simulated structures. In addition, the rate of extension of each organ is described by a logistic growth function, e. g. the beta growth function [27]. This function describes the dynamics of extension and biomass accumulation of organs: the application of this function to all organs over time then describes the growth of the whole plant.

The sink strength of a growing organ can be approximated by its potential growth rate, which can be described by the derivative of a growth function, e. g. the beta growth function proposed by Yin et al. [27]:

$$c_m = w_{max} \left( \frac{2t_e - t_m}{t_e(t_e - t_m)} \right)^{\left( \frac{t_m}{t_e - t_m} \right)} \quad (2.1)$$

where  $c_m$  is the maximum growth rate in the linear phase, i. e. at time  $t = t_m$ , and  $t_e$  is the time when growth ends (i. e. growth rate turns zero) as the maximum dimension or mass  $w_{max}$  is reached. The potential growth rate ( $ss_{pot}$ ) of an organ at time  $t$  is then computed as:

$$ss_{pot} = \frac{\delta w}{\delta t} = c_m \left( \frac{t_e - t}{t_e - t_m} \right) \left( \frac{t}{t_m} \right)^{\left( \frac{t_m}{t_e - t_m} \right)} \quad (2.2)$$

\* The method *getPgr()* is used to compute the potential growth rate in each organ.

Global sink demand is defined as the sum of all potential growth rates of concurrently growing organs, multiplied with the modelled step size:

$$sd_{tot} = \sum ss_{pot} \Delta t \quad (2.3)$$

Multiplication of  $ss_{rel}$  with the current size of the common assimilate pool  $cPool$  results in the realized growth rate  $gr_{real}$ , thereby making sure that this is not bigger than  $ss_{pot}$ :

$$gr_{real} = \frac{ss_{pot}}{sd_{tot}} cPool \quad (gr_{real} \leq ss_{pot}) \quad (2.4)$$

\* In the model this is implemented in each organ in the *getAGR()* method, where *cPool* is calculated using the method *getCentralPool()* of the associated individual.

Once growth of an organ takes place with rate  $gr_{real}$ , the central assimilate pool is updated accordingly.

Finally, a growth respiration term is considered in the form of a conversion factor ( $gglucoseg - 1dm$ ), which is proportional to growth rate as described in [7]. Maintenance respiration is computed by an organ specific fixed proportion of structural biomass [7]. Both terms are subtracted from the central pool at each step.

### 2.1.3 Vegetative and generative development

To simulate vegetative and generative development, a set of growth, developmental and branching rules is repetitively applied to a Bud module and all its ensuing organs, leading to the visible phenotype. The structural framework created thus is then used to simulate and analyse the dynamics of assimilate flow as dictated by local (potential) growth rates and assimilate availability in the central pool. A detailed description of the implemented rules is given in section 2.2.1.

Formation of a new organ from the Bud occurs after a certain time lag (plastochron). A new phytomer (consisting of internode and leaf) is formed with an initial length and diameter. The meristem is reinstated at the tip, its rank incremented by one and the plastochron set to zero. das muss dann aber im Modell auch noch implementiert werden... wir haben kein meristem, das macht bei uns alles das bud-organ

The potential extension and final dimension of organs (leaves, internodes, etc...) depends upon their rank and age, while the actually achieved dimensions are also a function of sink competition and assimilate availability, as described in section 2.1.2. Leaf dimensions are determined using, e.g., the beta growth function from [27], calculating dry matter increment as a function of time which is then converted into a scaling factor applied to an initial rectangle with a fixed ratio of width and length.

Once the generative stage is attained, flower formation takes place, followed by fruit formation according to a user defined fertilisation rate. As long as the central assimilate pool is not exhausted, and limited by their potential growth rate, fruits will grow.



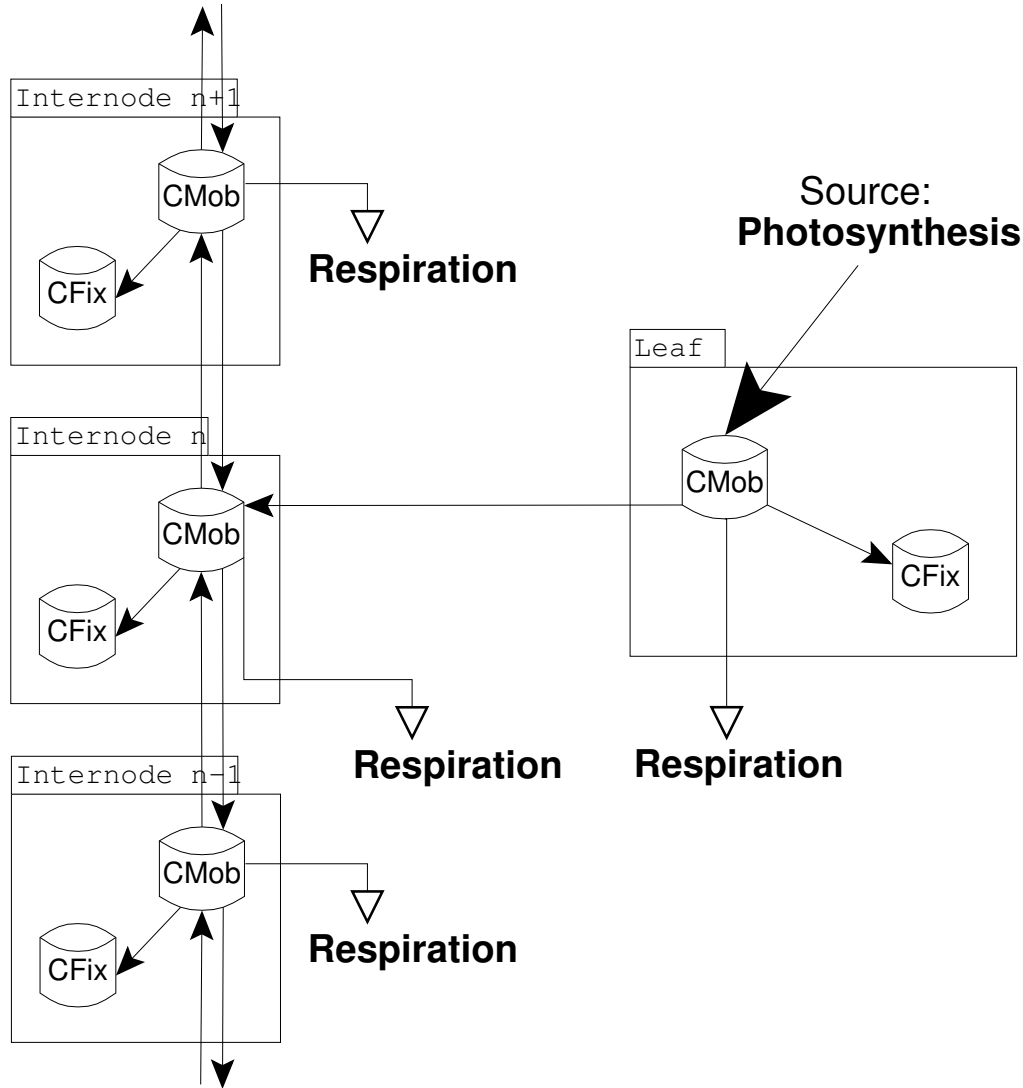


Figure 2.1: Reallocation of assimilates for growth as a function of relative sink strength of an organ, i.e. relative potential growth rate of that organ, divided by the sum of the potential growth rates of all competing organs

#### 2.1.4 Dynamics of the central assimilate pool

In the model, the central assimilate pool is updated as a function of local leaf photosynthesis or growth of an organ (Fig. 2.2). In the current setting, the dynamics of the central assimilate pool is characterized by tree phases: in the first, initial carbon is provided by the seed, which is rapidly exhausted during germination. Meanwhile, after unfolding of the first leaves, photosynthesis is started, thereby slowly filling up the central pool. During this phase, source and sink are in a balance and the pool is emptied at each step. If the central pool is not emptied after a step, the carbon will be stored in the individual for the next step.

Zwischen- bzw. Endspeicherung von Assimilaten geschieht unter bestimmten Umstaenden, also z.B. bei der Bildung von natuerlichen Vorratsspeichern wie Fruechten oder Zucker im Zuckerrohr, oder aber bei niedriger Temperatur kombiniert mit hoher Strahlung (hohes Source-

Sink-Verhaeltnis). Normalerweise erhoehrt die Pflanze aber bei einem zu hohen Source-Sink-Verhaeltnis entweder die Sink-Staerke (hoehere PGR eines einzelnen Sinks bzw. Bildung neuer Sinks durch Knospenaustrieb) oder verringert die Sourcestaerke durch Herunterregulieren der Photosynthese. Beide Mechanismen haben wir noch nicht eingebaut.

Platzhalter

Figure 2.2: Dynamics of carbon

### 2.1.5 Radiation model and light interception

The radiation model of GroIMP was used to simulate light distribution and local light interception. This model is based on a reversed path tracer algorithm with Monte-Carlo integration (Veach, [24]) and uses light sources and geometric objects constituting a so-called scene. The radiation model is invoked once per simulation step and is applied to the scene created and maintained within GroIMP (for details see Kniemeyer, [14]; Hemmerling et al., [10]). GroIMP provides several types of light sources like point light, spotlight, area light, or a sky object.

As a default the prototype uses a directional light source to simulate direct sun light whereas diffuse sky light is simulated using an array of 72 directional lights positioned regularly in a hemisphere in six circles with twelve lights each, with emitted power densities being a fixed function of the elevation angle (GH Buck-Sorlin, unpubl. res.). The light model is run with two parameters: total number of rays produced by all light sources in the scene, and the number of times a reflected or transmitted ray is traced.

## 2.2 File description

The model includes three main types of files, which will be described step by step below. In order to keep model structure as clear as possible all model parts are arranged in separate files, according to their functions.

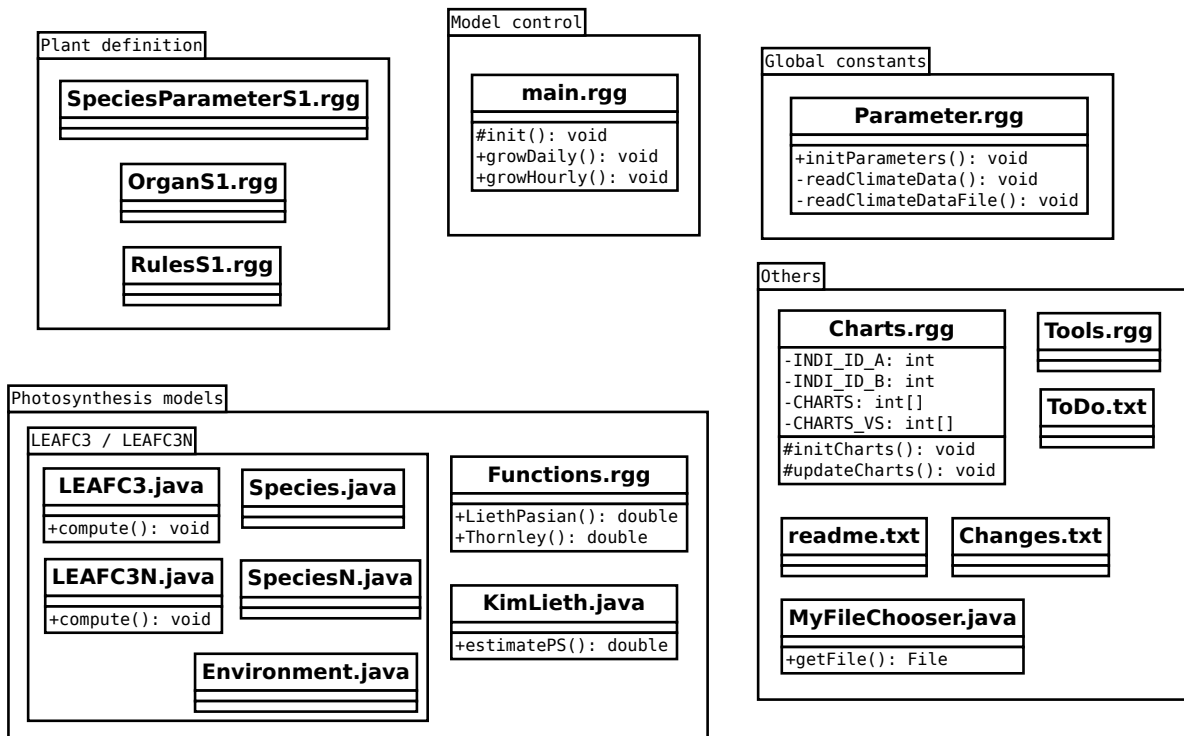


Figure 2.3: Model overview: Files separated by functionality

### 2.2.1 Model

These three files contain the main model. A parameter file contains global model parameters, while in a main file the model is initialized and controlled. All organ modules are defined in an organ file.

***Parameter.rgg*** Definition of global constants and all other model parameters.

***main.rgg*** Includes the main model definitions, derivation rules and growth functions.

***OrgansS1.rgg*** Declaration of all plant organs (modules) and constants of plant species one.

***RulesS1.rgg*** Declaration of all rules and constants of plant species one.

***SpeciesParameterS1.rgg*** Declaration of all species specific constants of plant species one.

#### Parameter.rgg

In this class important global variables such as climate data are declared. It is also possible to switch between climate data provided here and a file chooser mode to load external climate

data from files into the model. To keep it simple the file format of the climate file is rather restricted.

### Parameters:

**DEBUG\_MODE** Toggles between debug mode on and off to get some debug information during model development.

**BENCHMARK** Toggles between benchmark mode on and off to get information about one step. By default turned off.

**RUN\_MODI** Determines the (global) run modi of the model.  
Implemented run modi:

***DAILY = 0; HOURLY = 1;***

**RAYS** Number of rays used for the light model, default: 3.000.000.

**RADIATION** Light model, initialized with *RAYS* rays in case of daily and with *RAYS/24* for hourly run mode and recursion depth 7. (To get acceptable results, five million rays and a recursion depth of 10 are the minimum.)

**POW** Power density of global radiation: read from weather file or computed from method *calcPos()*, default: ? Einheit?

**DIFF\_POW** Power of diffuse sky light (computed from global radiation and fraction diffuse), default: 500 Einheit? (schon implementiert?)

**GRtoPPFDDL** Value for daylight, default: 4.5/2 Einheit?

**PHOTOSYNTHESIS\_MODEL** This parameter sets the (global) photosynthesis model which is used by all organs that carry out photosynthesis.

Implemented photosynthesis models:

***LEAFC3 = 0; LEAFC3N = 1; LIETHPASIAN = 2; KIMLIETH = 3;***  
***THORNLEY = 4;***

**MAX\_STEPS** Maximal number of steps the model can run, depending on the climate data available.

### Environment parameters:

**TEMPERATURE\_CLASS** Currently selected temperature class. Here the identification number of the *TEMPERATURECLASSES* array has to be used.

**TEMPERATURE\_CLASSES** Array of available temperature classes, identified by temperature in degree Celsius.

**LATITUDE** Latitude where the weather station is located. Used to calculate the correct sun position.

**START\_DAY** Day of the year when the model starts. Default: April 1

**TEMPERATURE** Two-dimensional array of temperatures.

**TEMPERATURE\_SUM** Two-dimensional array of temperature sums.

**TIME\_LENGTH** Total daily time of sun shine in  $[h]$ .

**RELATIVE\_HUMIDITY** Average daily relative humidity in percent in  $[0, 1]$ .

**GLOBAL\_RADIATION** Total daily radiation. Werden diese beiden arrays eigentlich richtig gebraucht? Ist dies die taegliche Strahlungssumme  $[MJ\ m^{-2}\ d^{-1}]$  oder die durchschnittliche Einstrahlung  $[W\ m^{-2}]$ ?? Wenn erstere, dann kann man sie nicht als Input fuer das PS-Modell gebrauchen, sondern nur fuer eine Berechnung der Radiation Use Efficiency (g Biomasse  $MJ^{-1}\ d^{-1}$ ). Im 2. Fall koennte man diesen Wert als Eingabe nehmen, aber es waere besser, ihn vorher mit Hilfe der Funktion `calcPos` in einen stuendlichen Wert umzurechnen.

**DIFFUSE\_RADIATION** Total daily diffuse radiation.

**DIFFUSE\_GLOBAL\_RADIATION** Fraction diffuse radiation.

**WINDSPEED** Average daily wind speed in  $[m/s]$

**CLOUD\_COVER** A average cloud cover.

**CO2\_CONCENTRATION** Constant average  $CO_2$  concentration.

For detailed information about the used an available climate data see chapter [C](#).

## Methods:

***initParameters()*** calls an other internal functions to initialise parameters. For example to load the climate data from file, if it is activated.

***readClimateData()*** It provides a file chooser dialogue or hard implemented file to loads climate data from an external file (by default: *climate.txt*). Take care that you use the right path to the climate file and the correct syntax for your system.

***readClimateDataFile*** implementation of the file parser for reading climate data from file.

## main.rgg

This file is the main file of the model. It includes the initialization, the rule part and the control flow of the model.

**Parameters:**

**step** number of simulated steps [days], current step size of the model.

**dayofyear** current day of year; initialized with the global parameter **START\_DAY** and incremented at each step;

**hourofday** Current hour of the model. In the daily run mode it is a constant, 12.

**startTime** Help variable for benchmark tests. Remembers the system time before a model step will be applied.

**deltaTime** Help variable for benchmark tests. Stores the time needed to carry out one entire model step.

**Methods:**

**init()** This method initializes the models with their variables and some additional items.

**initVariables()** initialize global counters, the charts and runs the *initParameters()* in the Parameter.rgg to load climate data from file.

**Light Part** Switch between three light conditions: a simple Light node (point light), a Sky object and a diffuse light sky object.

**Rotation Table** Only used in *makeImage()* to rotate the plant in an animation.

**Individual** Choice to initialize a single individual plant or a group of individuals.

Use

*Individual(id) Seed(INITIAL\_C, id)*

to get one individual plant or one of the loops to get an  $n \times m$  raster or a number of circles of individuals to simulate different planting densities.

**measureStick** Measure stick (to check plant sizes)

**Box** This is a square of  $1\text{ m}^2$  to check the amount of light reaching the ground.

**applyRules()** Calls the main control function for the model in class *RulesS1.rgg*.

**growDaily()** This function calculates a daily step of the whole model. It includes the global temperature update, some status and debug information, the call of the radiation model, the function call to apply plant rules, as well as the update of the charts if they are used.

**growHourly()** Works like *growDaily()* but in hourly run mode.

Note: It is not possible to combine the daily and hourly mode at once, because the light model will be not initialised new at each step.

It is at the developer's responsibility to make sure that the identification numbers of the individuals are unique. They do not have to be sequential or start at a defined point, so it is possible to start for example one species at 100 and a second species at 200 (e.g. in an intercropping model).

**Additional Methods:**

***manualHarvest()*** Harvest method triggered by the user, it will harvest all ripe fruits and produce a chart, if the harvest chart is activated in *Chart.rgg*.

***grow10Daily()*** Runs the *growDaily()* function ten times. Saves time (during development), because the update of the 3D view will be called only once after the ten steps instead of at each step. (To activate this function, change the visibility flag from private to public.)

***makeImage()*** Creates an image of the current scene, afterwards runs grow-function once; rotation of the plant can be switched off by setting the boolean flag rotation to false. As condition an inserted RotationTable-object at the base of the scene is needed. The default size of the produced image is 600x800 pixels, which can be changed easily in the code.

***calcLightOnGround()*** Method to test how much light is absorbed by the ground. To use it, a Box-object of one square meter has to be inserted into the scene. (Realistic values: between 400-500  $W/m^2$  or about 1800  $micromolPAR/m^2s$  or about 18  $MJ/m^2d$  in the field.)

**OrgansS1.rgg**

This class provides the object-oriented definitions for the plant organ modules and also a module to identify an individual plant. Basic organ constants are also defined in this class. This class is designed such that it is species-specific, which means all organs and nearly all constants relevant for one species are comprised here.

**Parameter (constants):**

The following parameters are implemented for all organ types.

$ORGAN \in \{SEED, ROOT, INTERNODE, BUD, LEAF, PEDUNCLE, FLOWER, FRUIT\}$

(Sollte nicht FRUIT zumindest einen Parameter numberOfSeeds haben?) du bist wieder beim raps oder ähnlichem und bei unserem generellen modell, lass es mal paprika sein, kommt es eher auf die anzahl der früchte an ;-). nee, im ernst würde ein numberOfSeeds nicht doch etwas weit gehen? und wenn wir diesen parameter einfuegen, auf welcher grundlage wollen wir ihn berechnen?

**ORGAN\_ID** Identification numbers, used e. g. to set the layer level of the organs.

**RATIO\_CARBON\_TO\_DRY\_WEIGHT\_ORGAN** Conversion ratio from carbon to dry weight.

**MAIN\_ORGAN** Maintenance coefficients of dry matter ( $g_{CH_2O}/g_{DMd}$ )

**ASRQ\_ORGAN** Assimilate requirements of dry matter ( $g_{CH_2O}/g_{DMd}$ )

**MAX\_AGE\_ORGAN** Maximal/total age of organs in degree-days ( $^{\circ}C$ ).

**FINISHED\_GROWING\_ORGAN** age of organs in degree-days after growth is finished ( $^{\circ}C$ ).

**FINAL\_DRY\_WEIGHT\_ORGAN** final dry weight of organ in  $g$  for  
 $ORGAN \in \{ROOT, BUD, PEDUNCLE, FLOWER, FRUIT\}$

**GERMINATE\_SEED** Time degree-days, in  $[^{\circ}C]$ , between sowing and germination.

**Q10MN** is the Q10 for maintenance respiration, i. e. the temperature dependence of maintenance respiration

**REFTMP** reference temperature for respiration and photosynthesis, usually  $25^{\circ}C$ .

**TEMPAIR** average daily air temperature

**ORGAN\_MAT\_NUMBER** shader definitions - organ textures - shader materials; These are the definitions of organ textures. The syntax of names is organ name\_MAT\_NUMBER.

## Methods:

***getPSImpl()*** implementation of the *getPS()* method. This method calculates the assimilate production per organ by using the currently selected photosynthesis model and run mode (daily or hourly). It simulates 24 steps (i. e. one per hour), summing up the hourly rates for the daily run mode.

This method has been implemented "globally" so that later photosynthesis can be easily invoked in other organs. (At present, photosynthesis is only implemented in the leaves.)

## Module descriptions:

### Organ - Interface

Interface that all plant organs have to implement. This facilitates writing of queries or rules relating to all organs.

## Methods:

***int getIndiID()*** returns the id of the individual to which the organ belongs

***int getAge()*** returns the age of the organ

***float getAbsorbedPower()*** in  $mol/organ_{area}$  (organ surface in  $m^2$ )



**float getDryWeight()** in  $[g]$

**float getFreshWeight()** in  $[g]$

**float getPGR()** potential growth rate [which unit?  $g\ h^{-1}$ ?]

**float getAGR()** actual growth rate [which unit?  $g\ h^{-1}$ ?]

**boolean isGrowing()** returns true if the organ is still growing

**float getMainTR()** Instantaneous maintenance respiration  $[mg_{CH_2O}/m^2s]$

**float getGrowthTR()** growth respiration  $[mg_{CH_2O}/m^2s]$

**void update(float power, float mean\_temperature)** updates organ-internal processes (e. g. incrementing organ age, dimension, geometry, or carrying out photosynthesis)

### Organ - Parameters

In all organs are the same private parameters implemented and some additional organ specific once. This are the parameter, which you can find in each organ:

#### Parameters:

**int age** organ age in  $[days/hours]$ , initializes with one.

**float absorbedPower** Absorbed power.

**float tempsum** Temperature sum.

**float dryWeight** Dry weight.

**Individual indi** The associated individual.

### Individual - Module

Represents a single individual plant and provides basic methods to get information about this plant.

#### Parameters:

**SHOW\_INFO** Using this flag the text label below the individual used to show information about the individual can be switched on or off.

**age** plant age, depending on current run mode:  $[days/hours]$ .

**centralCarbonPool** Central carbon pool  $[g]$ .

**sinkDemand** sink demand.

**sourceSinkRatio** source-sink ratio

**averageSourceSinkRatio** average source-sink ratio. (Average of ten steps.)

**tmpSumSourceSinkRatio** temporary variable to calculate the average source-sink ratio.

**psRegulationFactor** photosynthesis regulation factor.

### Methods:

Most of the methods are implemented in the following way

```

1 public float getValue() {
    return sum( (* x:Organ, (x.getIndiID()==indiID) *).getValue() );
3 }

```

These methods thus work with the help of XL graph queries, in this case the aggregate function sum sums up a value over all organs having the same ID as the individual.

**float getCentralPool()** Returns the central carbon pool of the plant, which is the sum of mobile carbon (assimilates) in all leaves of the plant.

**void getC(float value)** Subtracts value from central carbon pool and returns value in [g]

**float getDryWeight()** Returns the total dry weight of the plant.

**float getFreshWeight()** Returns the total fresh weight of the plant.

**float getMainTR()** Returns the instantaneous maintenance respiration.

**float getGrowthTR()** Returns the growth respiration.

**float getAbsorbedPower()** Returns the amount of absorbed power.

**float getLeafArea()** Returns the total leaf area of the plant.

**float getPS()** returns the amount of assimilates of the plant, which is the sum of mobile carbon (assimilates) in all leaves of the plant.

**float getSinkDemand()** Returns the sink demand.

**float getAverageSinkStrength()** Returns the average sink strength.

**float getSourceSinkRatio()** Returns the source-sink ratio.

**float getAverageSourceSinkRatio()** Returns the average source-sink ratio.

**int getNoOfGrowingOrgans()** Returns the number of growing organs.

*int getNoOfOrgans()* Returns the number of organs.

*int getNoOfShoots()* Returns the number of shoots.

#### Additional Methods:

*int getIndiID()* Returns the identification number (ID) of the plant.

*int getAge()* Returns the plant age in days or hours, depending on current run mode.

*void getOverview()* Prints an overview of all plant parameters.

#### ShootO - Module

Represents a single shoot and, like the module Individual, provides basic methods to obtain information about it.

#### Parameters:

**SHOW\_INFO** Using this flag the text label below the individual used to show information about the individual can be switched on or off.

#### Methods:

*int getIndiID()* returns the identification number (ID) of this plant.

*int getNoOfGrowingOrgans()* returns the number of growing organs.

*int getNoOfOrgans()* returns the number of organs.

*float getLeafArea()* returns the leaf area of this shoot.

All other organ definitions are implementations of the Organ interface, thus for the method description see section [Organ interface](#).

#### Seed

#### Additional Parameters:

**germinating** Indicates if the seed is germinating or not.

**Additional Methods:**

***void decWeight(float value)*** decreased the amount of carbon. It will be only called by the Individual during the first steps until the local carbon pool is not empty. gbs: verstehe nicht, was diese Methode tut. im ersten modellschritt hole ich vom samen den C und legen ihn auf den zentralen pool im individuum, er wird aber im samen nicht geloescht, sondern schrittweise entsprechend des verbrauchs verringert. und genau dieses schrittweise verringern macht diese methode, sie wird also vom indi aus aufgerufen.

***boolean isDead()*** Indicates if the seed is dead.

***boolean isGerminateConditions()*** Returns true if the seed has collected enough temperature to germinate and it is not already germinated.

**Implementation details:**

**Geometry** is implemented as a Sphere object.

**Initialization** The organ is initialized with an initial amount of carbon. For this the constant **INITIAL\_C** is used.

**Root****Additional Parameters:**

**none**

**Additional Methods:**

***finalDryWeight()*** calculates the final dry weight of the total root system depending only on **tempsum**. For the calculation a Beta growth function (see section 2.2.3) is parametrized to get a final weight of **FINAL\_DRY\_WEIGHT\_ROOT** *g* attained at an age of **FINISHED\_GROWING\_ROOT** degree days.

**Implementation details:**

**Geometry** is implemented as a Sphere object.

**Bud**

**Additional Parameters:**

**sensedPower** stores the actual amount of absorbed radiation.

**tmpSensedPower** stores the amount of absorbed radiation in order to determine when a new internode can be formed.

**plastochron** Distance (in degree-days) between the formation/appearance of two leaves. It will be initialized with a value according to this  $60 + ((rank < 10)?0 : 6 * rank)$ ; formula and decreased at each step into the *update()* function by the current mean temperature.

**Additional Methods:**

**int maxRank()** Calculates the maximum rank of a bud and thus determines the maximal number of phytomers per shoot.

**finalDryWeight()** calculates the final dry weight of the organ depending only on **tempsum**. For the calculation a Beta growth function (see section 2.2.3) is parametrized to get a final weight of **FINAL\_DRY\_WEIGHT\_BUD** *g* attained at an temperature sum of **FINISHED\_GROWING\_BUD** degree days.

**float getSensedPower()** Returns the actual amount of absorbed radiation.

**boolean isBranchingConditions()** Returns true as long as the branching conditions are suitable. By default, if *rank* ≤ 4. Thus, currently the formation of side shoots is just determined by topological parameters.

**boolean isFloweringConditions()** Returns true as long as the conditions to build flower are suitable. By default, if *rank* ≥ 11 && *rank* ≤ *maxRank()*.

**boolean isGrowingConditions()** Returns true as long as the growing conditions are suitable. Formation of a new organ from a bud occurs after some resting time (plastochron). By default, if *rank* < 11 && *order* ≤ 2 and the current amount of sensed radiation is bigger than a threshold, currently 0.022 komischer Wert - ist ja auch frei erfunden. As a result, a new internode will be that depending on the amount of light, under optimal condition (initialised light model with 5.000.000 rays and a reflection depth of 10), just every 4 to 6 days build. nachsehen, ob die tagesangaben noch stimmen

**Implementation details:**

**Geometry** is implemented as sphere object.

**Internode**

**Additional Parameters:****none****Additional Methods:**

***finalDryWeight()*** calculates the final dry weight of an internode depending on the rank.

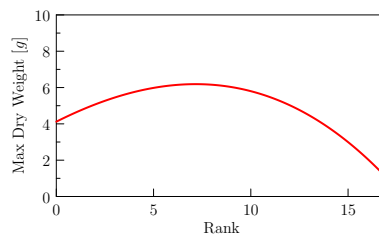


Figure 2.4: Maximal weight per rank (used function:  $a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3$ )

***float getArea()*** Returns the surface area in  $m^2$

***float getVolume()*** Returns the volume of this organ in  $m^3$

***float getAbsorbedPowerM2()*** absorbed power projected to one square meter organ surface area.

***updateGeometry()*** Updates the organ geometry according to the current dry weight.

**Implementation details:**

**Geometry** is implemented as Cylinder object.

**Texture** a simple green image as shown in Fig. D.3 was used.

**Leaf****Additional Parameters:**

**localCarbonPool** local carbon pool. Each step, thirty percent of the assimilate will be stored directly to this local pool.

**myShader** The shader used to get the leaf texture.

**image** Additional variable only used if the senescence is turned on.

**scaleFactor** current scaling factor. This factor will be updated each step and will be applied to the initial leaf of a fixed size to model leaf growth. [0, ..]

**SENESCENCE** turns the alpha-blending, between an initial green leaf and an old yellow one, on and off to visualize senescence.

### Additional Methods:

**getLength()** returns the length of the leaf. It is calculated as product of the actual scaling factor  $scaleFactor * 100$  and the initial constant length of 0.001 m. (In this model the ratio between leaf length and width is assumed to be fixed.) [m]

**getWidth()** returns the width of the leaf. It is calculated as product of the actual scaling factor  $scaleFactor * 100$  and the initial constant width of 0.00036 m. (In this model the ratio between leaf length and width is assumed to be fixed.) [m]

**finalDryWeight()** calculates the final dry weight of a leaf depending on rank.

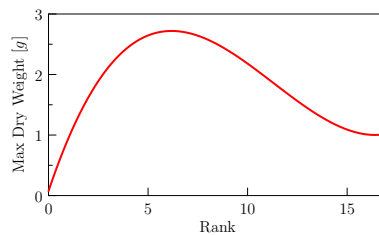


Figure 2.5: Maximal weight per rank (used function:  $a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3$ )

**float getAbsorbedPowerM2()** absorbed power projected to one square meter organ surface area.

**getArea()** returns the leaf area as product of *getLength()* and *getWidth()* [m<sup>2</sup>] nicht "mal Formfaktor"??

**boolean isDead()** Indicates if the seed is dead.

**getPS()** This function calls the global *getPSImpl()* (implementation) method. It is used to calculate assimilate production per leaf by using the actual selected photosynthesis model and run mode. Called from the internal function *updateCMob()* once every step.

**float getPAR()** calculates the photosynthetically active radiation based on the absorbed power *getAbsorbedPower()*.

**updateGeometry()** Updates the organ geometry according to the current dry weight.

**void updateShader()** this function is called if the senescence is turned on (**SENESCENCE = true**). As long as the leaf is growing it calculates every 5th step a new shader using an additional function (*blendImg()* in *Tools.rgg*) for alpha-blending between two images.

**Implementation details:**

**Geometry** is a combination of a cylinder object as peduncle and a leaf object is used for the leaf blade with a small rotation in-between to slightly bend the leaf blade. An additional rotation node at the base of the peduncle bends the leaf using a function depending on age to compute the bending angle.

The size is based on the current dry weight of the organ from which first the area will be calculated. For simplicity, a constant length - width ratio was chose, allowing to simply scale the leaf object (initialised with this fixed ratio) during growth. The variable *scalingFactor*, which scales the geometry, will be updated with the actual growth rate.

**Texture** is a leaf image with transparent background. If senescence is turned on, a new shader will be calculated every fifth step by using an alpha-blend between a juvenile, green (Fig. D.1) and an old, yellow adult leaf image (Fig. D.2).

**Senescence** is not considered in the normal model mode . The implemented method of continuous alpha-blending between two images (one each for the juvenile and the adult stage) is, unfortunately, too time-consuming for use during model development, but can be used in the final stage to get an optimal visual result. To activate it set the organ constant **SENESCENCE** from *false* to *true*. ab hier habe ich die Kleinigkeiten aus Zeitgruenden nicht mehr markiert. Die Visualisierung der Seneszenz ist m.E. nicht wichtig, es sei denn, man betont, dass alte Blaetter ein anderes Absorptionsspektrum haben.

**Peduncle****Additional Parameters:**

none

**Additional Methods:**

*updateGeometry()* Updates the organ geometry according to the current dry weight.

**Implementation details:**

**Geometry** is implemented as sequence of five cylinder objects to get at slightly bent shape. To simulate bending during ageing, the peduncle object searches for its predecessor object, which is a rotating node and resets its angle. To calculate the bending angle an age-dependent function is used.

**Texture** Same texture as for internodes is used (Fig. D.3).



## Flower

### Additional Parameters:

**none**

### Additional Methods:

*boolean isDead()* Indicates if the seed is dead.

### Implementation details:

**Geometry** is implemented as Sphere object.

**Texture** is a colour gradient from green to white depending on organ age, so the final stage is a white flower.

## Fruit

### Additional Parameters:

**geometryTMP** Temporary variable to generate the geometry of the fruit.

**cut** Indicates if the fruit is cutted or not.

### Additional Methods:

*boolean isRipe()* Indicates if the fruit is ripe or not.

*boolean isCut()* Indicates if the fruit is cutted.

*void setCut()* Sets the cut flag of this fruit.

*updateGeometry()* Updates the organ geometry according to the current dry weight.

### Implementation details:

**Geometry** is implemented as simple sphere object.

**Senescence** is a colour gradient from green to red depending on organ age.

## RulesS1.rgg

In this class all rules referring to one plant species are collected.

### Methods:

***applyRules()*** This function is the main control function of the model. For reasons of better housekeeping it is divided into five parts (each implemented in a separate method):

```

1 morphologyRules();
  cutRules();
3 //transportRules();
  organUpdates();
5 otherRules();

```

***morphologyRules()*** Build-up of plant morphology. These are the main rules for organ formation.

```

1  s:Seed, (s.isGerminateConditions()) ==>
    Root(s.getIndiID()) s Bud(1,1, s.getIndiID());
3
5  b:Bud(rank, order, indiID), (b.isGrowingConditions()) ==>
    RV(-0.15)
    Internode(rank, order, indiID)
    [Leaf(rank, order, indiID)]
7    if(b.isBranchingConditions()) (
9      [RL(30) Shoot0(indiID) Bud(rank+1, order+1, indiID)]
    )
11   Rotate(random(-5,5), random(-5,5), PHYLLOTAX + random(-5, 5))
    Bud(rank+1, order, indiID);
13
15  b:Bud(rank, order, indiID), (b.isFloweringConditions()) ==>
    Internode(rank, order, indiID)
    [Leaf(rank, order, indiID)]
17  [
    RL(START_PEDUNCLE_ANGLE) Peduncle(rank, order, indiID)
19    [Flower(rank, order, indiID)]
    if(order==1 && probability(CHANCE_TO_POLLINATE_MAIN)) (
21      [Fruit(rank, order, indiID)]
    )
23    if(order==2 && probability(CHANCE_TO_POLLINATE_SIDE)) (
    [Fruit(rank, order, indiID)]
25  )
  ]
27  Rotate(random(-5,5), random(-5,5), PHYLLOTAX + random(-5, 5))
    Bud(rank+1, order, indiID);

```

The conditions for the formation of a new organ are integrated in the organ itself. See the Bud definition in section 2.2.1 for further details.

**cutRules()** Removal of leaves or flowers when they have reached their final age.

```

2  // if the carbon resource is empty
x:Seed, (x.isDead()) ==> ;

4  x:Flower, (x.isDead() && probability(0.88)) ==> ;
x:Leaf, (x.isDead() && probability(0.5)) ==> ;

6

8  // peduncle without a flower or fruit
x:Peduncle, (x.getSuccessor()==null && x.getBranch()==null) ==> ;

10 // ripe fruits
/*
12 x:Peduncle [y:Fruit], (!y.isGrowing() && probability(0.5) && !y.isCut()) ==>
    {
14     Tuple3d t = location(y);
        y.setCut();
16     }
    x, ~ Null(2.5*t.x, 2.5*t.y, 0) y;
18 */

```

**transportRules()** Predefined transport rules to model exchange between the organs. Currently not used.

```

2  // internode <--> leaf
x:Internode [y:Leaf] ::> {
    // -->
4    y[yourParameter] += x.get?();
    // <--
6    x[yourParameter] += y.get?();
    }
8  // internode <--> internode
x:Internode (-->)+ y:Internode ::> {
10    // -->
    y[yourParameter] += x.get?();
12    // <--
    x[yourParameter] += y.get?();
14    }

```

**organUpdates()** Update of state variables in existing plant organs (e.g. *setAbsorbedPower()*).

```

2  // for all organs
x:Organ ::> {
    // update the organ (e.g. geometry and/or photosynthesis)
4    x.update(RADIATION.getAbsorbedPower(x).integrate(),
        (RUN_MODI==DAILY)?TEMPERATURE[TEMPERATURE_CLASS] [dayofyear] :
6        TEMPERATURE[TEMPERATURE_CLASS] [dayofyear]/24f);

```

```

8      if (x instanceof Bud) {
          // update the sensed power of the bud
10      ((Bud)x).setSensedPower(RADIATION.getSensedIrradiance(x).integrate());
          }
12  }

14  x:Individual ::> {
          x.update();
16  }

```

*otherRules()* For debugging or statistical output.

```

      x:Individual, (x.getIndiID()==0) ::>
2      println(" central Pool = "+x.getCentralPool()+"g C");

4 // x:Individual, (x.getAge(>15) ::> x.getOverview();

```

### SpeciesParameterS1.rgg

In this class all plant constants belonging to one plant species are collected.

#### Parameter (constants):

**PHYLLOTAX** is the phyllotactic angle between two leaves.

**INITIAL\_C** initial carbon amount of the plant. (Only used during initialisation of a Seed object.)

**CHANCE\_TO\_POLLINATE\_MAIN** describes the probability of a main stem flower to be pollinated.

**CHANCE\_TO\_POLLINATE\_SIDE** describes the probability of a side-shoot flower to be pollinated.

**START\_PEDUNCLE\_ANGLE** the initial angle of a peduncle.

**TODO** TODO

### 2.2.2 Photosynthesis models

The FSPM-Prototype provides two types of the LEAFC3 photosynthesis model [21] (with [?] and without consideration of nitrogen [20]), an implementation of the Kim-Lieth model [12] as well as models based on simple light-response curves like the the Lieth-Pasian model [?] and the Thornley model [23].

**Files:**

***LEAFC3.java*** Java implementation of Nikolov photosynthesis model for leaves of C3 plants.

***Species.java*** Species-specific model parameters for Nikolov photosynthesis model for leaves of C3 plants.

***Environment.java*** Environmental model parameters for Nikolov photosynthesis model for leaves of C3 plants.

***LEAFC3N.java*** Java implementation of Nikolov photosynthesis model for leaves of C3 plants with nitrogen.

***SpeciesN.java*** Species-specific model parameters for Nikolov photosynthesis model for leaves of C3 plants with nitrogen-sensitivity.

***KimLieth.java*** Photosynthesis model by Kim and Lieth [12], parameterized for cut-rose

***Functions.rgg*** Functional models and process-based implementations. (Lieth and Pasion [19]; Thornley [23], Pien 2007 [?], Eq. 7.3)

**Description:****General**

To estimate local light interception and leaf photosynthesis, the radiation model of GroIMP was used. It is described in detail in [10]: The method is based on an inversed Monte-Carlo raytracer [24] and uses light sources and geometric objects placed into a scene. We used a single point light source with a power of  $600 \text{ W/m}^2$  (corresponding to incident global radiation,  $1367 \text{ W/m}^2$ , attenuated by the atmosphere). To use the light model, an instance of it is invoked with two parameters: the total number of rays produced and traced from directly lit geometric objects to all the light sources in the scene, and the number of times a reflected or transmitted ray will be traced. To produce acceptable results, the light model has to be initialized at least with 5 million rays and a recursion depth of 10.

Leaf blades are modelled as parallelogram objects of different size and orientation. Each leaf has a parameter *absorbedPower* which stores the amount of light absorbed by the leaf [ $\text{W/leafArea}$ ]. To convert *absorbedPower* to intercepted PAR (Photosynthetically Active Radiation), the private function *getPAR()* will be called.

*Once a leaf is formed, it is identified with a label and its absorbed PAR determined using a method *getAbsorbedPower()* of the light model, which returns the spectrum of absorbed PAR. This spectrum is converted to Photosynthetic Photon Flux Density [ $\mu\text{molPPFD/m}^2\text{s}$ ] using a coefficient of 2.27 for daylight.*

The *getPSDaily()* method which invokes the currently selected photosynthesis model, is used to calculate daily assimilate production per leaf. The output of all leaves is at each daily step added up to a central assimilate pool.

Since only daily totals of global radiation were available, the expected value for a given hour of the day was estimated using a sine function described in [7], assuming a daily average of atmospheric transmissivity, estimated as  $1 - (s/d)$  where  $s$  is the number of sunshine hours  $[h]$  and  $d$  the daylength  $[h]$ , from civil dawn to civil dusk]. bin mir nicht mehr sicher, ob das immer noch stimmt...

### LEAFC3 / -N

The implemented LEAFC3 and LEAFC3N models are Java-adaptations of the revised version of the Delphi Pascal code from Nikolov 1995 [21].

LEAFC3 is a generic model for the estimation of short-term steady-state fluxes of  $CO_2$ , water vapour, and heat from leaves of C3 plant species, explicitly coupling all major processes involved in photosynthesis (biochemistry of assimilation process, stomatal conductance, leaf energy balance). This model has already been successfully used to model gas exchange in wheat by [20], and in barley (Ole Kniermeyer and Gerhard Buck-Sorlin, unpublished).

For the LEAFC3 and LEAFC3N model are climate data necessary as input. This data are defined in the *Parameter.rgg* file or can be loaded from climate file (*climate.txt*), they containing daily values of mean temperature, global radiation, and relative humidity and mean cloud density and  $CO_2$  concentration. See the appendix C for further informations about which data are necessary and the syntax of the climate file.

The net assimilation rate ( $An$ ,  $\mu mol_{CO_2}/m^2s$ ) is the only used output of the FSPM-P, but there are also other values calculated like  $g_{sv}$ ,  $g_{bv}$ ,  $Tl$  and  $LHeat$ .

### Kim and Lieth

- Java implementation: TODO

### Lieth and Pasian

- Java implementation: TODO

### Thornley

- a simple light-response curves based on Thornley model [23].: TODO

### 2.2.3 Other

This class of files contains some useful tools for statistical analysis, some mathematical tools and an input class to read in climate data from a file.

**Charts.rgg** Additional class for statistical plots.

**Tools.rgg** Collection of useful tools and mathematical functions, e.g. for growth.

**readme.txt** Readme file with some short information.

**MyFileChooser.java** Class provides a simple FileChooser to load external files.

**ToDo.txt** To do list.

**Changes.txt** Information about changes / new features / updates, mainly for developers.

#### Charts.rgg

In this class fifteen predefined charts are implemented presenting the time course of various simulated state variable. There are two classes of plots: one to monitor a single individual and the second to compare two individuals. With the parameter **INDI\_ID\_A** the individual to be monitored in the "single mode" can be selected by its identification number. To pick charts from the **CHART\_DESCRIPTION** list, simply add the id of the selected charts to the **CHARTS** array.

To compare two individuals *A* and *B* set their id's to **INDI\_ID\_A** and **INDI\_ID\_B** parameters, select the charts you want from the **CHART\_DESCRIPTION\_VS** list and add the id's of the selected charts to the **CHARTS\_VS** array.

Make sure that no more than ten charts in total are selected ( $\max(|\mathbf{CHARTS}| + |\mathbf{CHARTS\_VS}|) = 10$ )! If you need more than ten charts you have to add more **DatasetRef** to the array of **DATASETS**.

**INDI\_ID\_A** Identifies the individual from which the charts are made.

**INDI\_ID\_B** Identifies the second individual to be compared with the first one.

**CHARTS** Array of indices of the charts to be produced for monitoring a single individual (each index has to be one of **CHART\_DESCRIPTION**).

**CHARTS\_VS** Array of indices of the charts to compare two individuals (each index has to be one of **CHART\_DESCRIPTION\_VS**).

#### Tools.rgg

This class provides module definitions, mathematical functions and some additional functions.

**Function:**

***updateLight()*** Updates the power of diffuse radiation and the power and position of the sun object.

```

2  // diffuse
   dls:DiffLightSky ::> {
4    dls.update(hourofday, dayofyear);
   }
   // direct
6  sn:Sun ::> {
   sn.update(hourofday, dayofyear);
8  }
```

***double[] calcPos(float latitude, int doy, int time)*** Function to calculate  $x, y, z$  of solar position, instantaneous and daily power, and day length at any time and day of the year; From a model by Jan Goudriaan, obtained 14.2.2008 [?]

**Modules:**

***module measureStick(int steps, float stepsize)*** measure stick to have a visual comparison of a simulated structure with a standard length  
parameter: steps - number of steps; stepsize - length of one step (in meter)

***DiffLightSky(int time, int dayofyear)*** Diffuse sky light object, consisting of 72 directional light sources arranged in a hemisphere in 6 horizontal layers with 12 equally spread lights each.  
parameters: doy: day of year (1 - 365), time: hour of day (0 - 23), power: overall measured power of diffuse radiation.

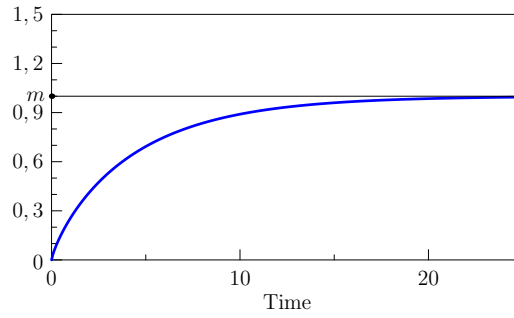
***Sun(int time, int dayofyear)***

***DiffSingLight(float power)***

**Mathematical functions:** All mathematical functions are implemented according to the same schema, the input parameters are always of type double *time* and have a double array *parameter* with the function parameters. They return a double value which is the evaluation of the specific functions during the given parameter at the entered time. The additional parameter *y0* represents a base value, which is added to the result.

***evaluateChapmanRichards(...)*** Chapman-Richards growth function  
function:  $h(time) := m(1 - \exp(-k * time))^n + y0$   
parameters:  $[m, k, n, y0]$  ( $m$ =asymthote)

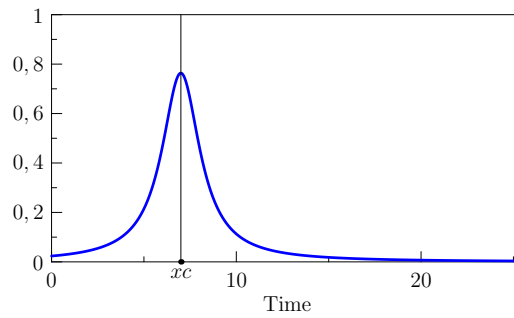




***evaluateLorentz(...)*** Lorentz function

function:  $h(time) := 2 * A / \pi * w / (4 * (time - xc)^2 + w^2) + y0$

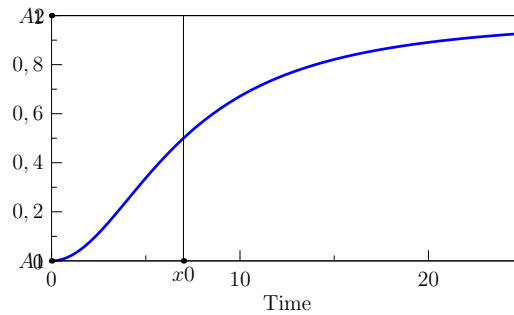
parameters:  $[A, xc, w, y0]$



***evaluateLogistic(...)*** Logistic function

function:  $h(time) := A2 + (A1 - A2) / (1 + (time/x0)^p) + y0$

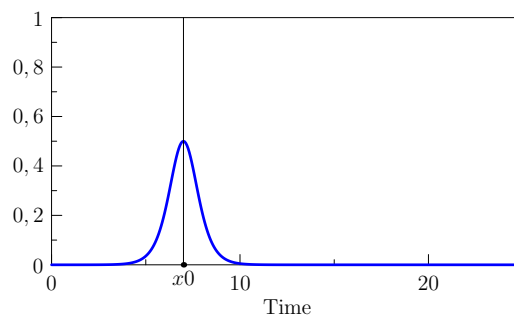
parameters:  $[A1, A2, x0, p, y0]$



***evaluateLogisticDerivative(...)*** Derivative of logistic function

function:  $h(time) := (p * A2 * \exp(-p * (x - x0))) / (\exp(-p * (x - x0)) + 1)^2 + y0$

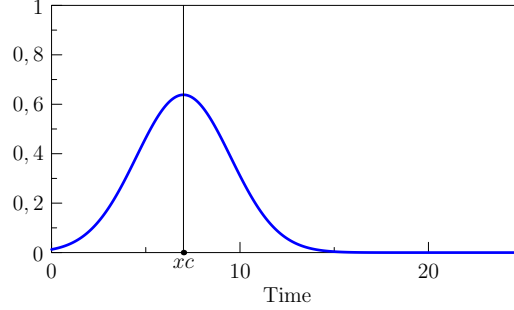
parameters:  $[A2, x0, p, y0]$



***evaluateGauss(...)*** Gauss function

function:  $h(time) := A * \sqrt{2/PI} / w * \exp(-2 * ((x - xc)/w)^2) + y0$

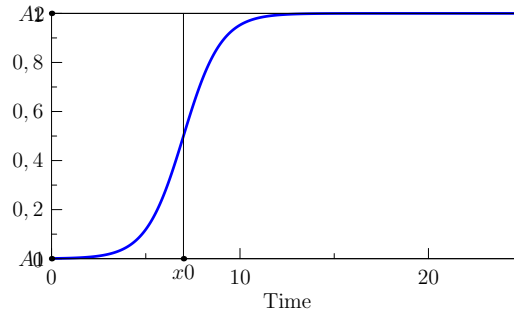
parameters: parameters  $[A, xc, w, y0]$



***evaluateBoltzmann(...)*** Boltzmann function

function:  $h(time) := A2 + (A1 - A2) / (1 + \exp((x - x0)/dx)) + y0$

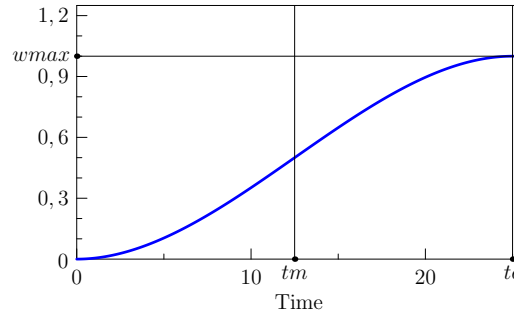
parameters:  $[A1, A2, x0, dx, y0]$



***evaluateBeta(...)*** Beta function, after Yin [27]

function:  $h(time) := wmax * (1 + (te - time)/(te - tm)) * (time/te)^{(te/(te-tm))}$

parameters:  $[wmax, te, tm]$

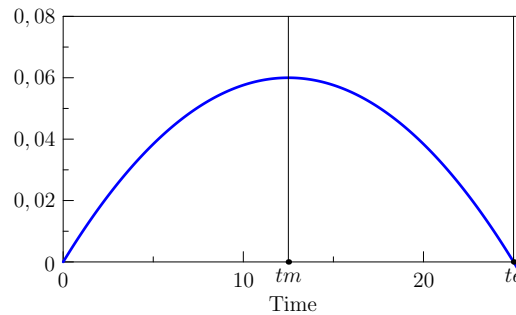


***evaluateBetaDerivative(...)*** Derivative of Beta function (growth rate), after Yin [27]

function:  $cm = wmax * ((2 * te - tm)/(te * (te - tm))) * (tm/te) * (tm/(te - tm))$

$h(time) := cm * ((te - x)/(te - tm)) * (x/tm) * (tm/(te - tm))$

parameters:  $[wmax, te, tm]$



### Additional functions:

***printOutToFile()*** An example how to write data into a file. You can use this function to document data to use it in other software.

***blendImg()*** implements a simple alpha-blending between two images. It is used to simulate leaf senescence using two leaves representing a juvenile and an old mature state.

### Readme.txt

File with some short information. A list of files that belong to this project.

### MyFileChooser.java

Class provides a simple FileChooser to load external files. Until now it is just used to read climate data from file.

### ToDo.txt

A todo list with things we want to implement and/or change in future. It also includes a list of wishes and for helpful suggestions as well as bugs and known problems.

### Changes.log

Primary for information exchange between developers about code changes.



# Chapter 3

## Building A New Model

How to create a species-specific FSPM based on FSPM-Prototype

Steps:

1. Systems analysis and definition of modelling targets (What should be the output of the model? Which input is necessary / facultative?)
2. Data acquisition and preparation, using the measurement protocol and time schedule.
3. Data analysis to obtain necessary model parameters as described in the chapter on data processing.
4. Implementation of an initial backbone model of structural development using derivation rules (L-Systems).
5. Reparameterization and calibration of the FSPM-Prototype
6. Model testing (e.g. sensitivity analysis, scenario studies, . . . )
7. Model validation using an independent data set

Types of measured parameters:

- Biometry (length, diameter of internodes and leaves, at least every three days, see table)
- Architecture (once or twice during the growing period, intermediate & final state)
- Phenology (ideally daily or every other day)
- Photosynthesis (once or twice during the growing period, on leaves of different ages and positions)
- Microclimate (continuously: local temperature with temp. sensors)
- SPAD measurements at harvest gbs: die Benutzung eines SPAD-Meters haben wir nicht explizit beschrieben, oder? NEIN, was ist ein SPAD?
- Chlorophyll fluorescence (currently not directly processed in the model)

Data types for model parameterisation:

- Biomass
- Biometry (metric and meristic)
- Architecture
- Phenology
- Photosynthesis
- Environmental data
- Planting density

The method to extract model parameters from measured parameters is described in chapter 8 - Data Processing.

### 3.1 Single plant model

Species-specific parameters and rules are stated in two files, *OrganS1.rgg* and *RulesS1.rgg*. In *OrganS1.rgg* S1 represents species one; here all organs and the main part of global parameters are defined. Growth and development rules and some other global constants are summarised in *RulesS1.rgg*.

### 3.2 Multiple species

The infrastructure of the FSPM-P allows modelling of more than one species in a convenient way. Because of the modularised implementation there are only two files including everything for a single species definition, see the description of a "Single plant model" above.

To implement a multiple species model the following steps should be applied:

1. Develop each single-plant model separately (preferably based on the FSPM-P).
2. Calibrate each model separately, combine them in one model.
3. Combining the two models (A and B): Ich glaube, dass diese Anleitung eher abschreckend wirkt. In der Praxis muss man es natuerlich genauso machen, wie Du es hier beschreibst. Aber langfristig waere es doch besser, das Kombinieren von Modellen eher mit einer Art Drag und Drop zu machen, so wie bei OpenALEA(?)
  - (a) Extract models A and B on your desktop with a zip program. (gsz-files are just zip files)
  - (b) Take a "new" FSPM-P as base.
  - (c) Copy the *OrgansS1.rgg* and *RulesS1.rgg* from model A into the species files of the "new" prototype.

- (d) Use the GroIMP "File Explorer" dialogue to add the files *OrgansS2.rgg* and *RulesS2.rgg* from your extracted model to your new model. (**Object** → **New** → **Add File**)
  - (e) Add all shaders and images to the new model. Take your extracted models as source.
  - (f) Add the imports of *OrgansS2.\**; and *RulesS2.\**; to all files where the files of species one are also imported.
  - (g) To eliminate conflicts with the now doubled definition of organs, rename all modules in *OrgansS2.rgg* like **Organ** to **OrganS2** and **Root** to **RootS2** and so on.
  - (h) Write a new initialisation for your multi-species model. For example, you can use the predefined loops to create a  $n \times m$  raster, in which several specimens of each species are placed in alternating rows by adding a condition like *if( $i \% 2 == 0$ ) initS1(); else initS2();*.
  - (i) Save your new model and start to remove occurring errors. damit geben wir natuerlich zu, dass die obige Anleitung schiefgehen koennte/ wird, oder?
4. Until now, it was more the "technical" part. In order to model interaction, like sensitive reactions, between the two species you need to implement special rules for that.

### 3.3 Experiments to try

There are several aspects, questions and objectives connected with building an FSPM. In this section some of the common objectives are explained exemplarily, such as:

- Quantitative yield prediction
- Decision-support and consultancy
- Education and teaching
- Research platform (e. g. investigating physiological mechanisms leading to complex crop phenotypes)
- Design of new plant types (e. g. ideotypes, adapted to new, more extreme climates, or for yield optimisation under given climate conditions)
- 

#### 3.3.1 Environment conditions

- Climate conditions
  - Changing **single climate parameters**: Try another climate data set, e. g. with a higher mean temperature or change one of the other climate parameters like global radiation.
  - Different **seeding date**: What is the effect of variation in seedling emergence? This can be explored by changing the global parameter **START\_DAY**.

- Planting density  
Explore the effect of changing plant density and/or plant arrangements to see the effect on inter-individual competition for light.

### 3.3.2 Model conditions

- Transport rates  
Changing internal **transport rates** to see ...
- Morphological changes  
Possible scenarios:
  - Change growth rates or maximal sizes of leaves
  - Modify pollination rate



# Chapter 4

## Experimental Settings

Before the start of an experiment, experimental conditions need to be defined.

It is not the purpose of this manual to give advice on suitable experimental conditions. However, our aim is to point the user to the importance of obtaining a dataset that is suitable for subsequent use in an FSPM. For more information, see van der Heijden et al. 2007 G.W.A.M. van Der Heijden, P.H.B. de Visser and E. Heuvelink (2007): Measurements For Functional-Structural Crop Models. In: J. Vos, L.F.M. Marcelis, P.H.B. de Visser, P.C. Struik and J.B Evers (eds.), Functional-Structural Plant Modelling in Crop Production, pp. 13-25. Chap. 22 For our purposes this essentially means three things: record the initial environmental settings; make sure they are uniform for all experimental plants/plots; determine the number of plants to be grown and used for measurements.

### 4.1 General

In order to get comparable results pay attention to:

- Uniform soil conditions: use same type of soil for all plants
- Uniform seed material (especially with respect to age and size). If necessary, conduct a germination test.
- Planting densities: choose a planting density that favours uniform germination and seedling establishment. If necessary reduce plant density by thinning or replanting. Too high plant density will increase seedling mortality and reduce branching, too low plant density leads to too much branching and thus untypical plants, too.
-

## 4.2 Single Plant Model

TODO ...

Growing conditions:

- Controlled climate (greenhouse, climate chamber)
- In field
- Number of plants: 40 (+10) plants
- Time period: 2009-12-01 - 2010-01-31
- Seeding: 2009-10-01

Only for greenhouse and climate chamber (limited for room):

- Temperature:  $25^{\circ}C$
- Humidity: 75%
- $CO_2$  concentration:  $0.000375 \text{ mol}_{CO_2}/\text{mol}_{air}$

## 4.3 Temperature Sensitive Model

The following is an example of an experimental setting to obtain parameters for a temperature sensitive model. In this setting, plants are grown at three temperature classes, a low, an optimal and a high one and as reference some field-grown plants. In this way, for instance the base temperature for growth can be obtained and then the model be run on a temperature-sum scale instead of time. For the low- and high-temperature treatments a climate chamber is the best choice to get controlled conditions and for optimal growing conditions (at least in winter) you may want to use a greenhouse.

- 2 climate chambers and greenhouse:
  - Number of temperature treatments: 3
    - Low temperature treatment (climate chamber) :  $10^{\circ}C - 12^{\circ}C - 14^{\circ}C$
    - Optimum temperature treatment (greenhouse) :  $18^{\circ}C - 20^{\circ}C - 22^{\circ}C$
    - High temperature treatment (climate chamber) :  $26^{\circ}C - 28^{\circ}C - 30^{\circ}C$
  - Humidity: 75%
  - Number of plants measured: 40 (+5) per temperature treatment (total: 120 plants)
  - Time period: 2009-12-01 - 2010-01-31
  - Seeding: 2009-12-01
- Field:
  - Number of plants measured: 40 (+10) plants
  - Time period: 2009-12-01 - 2010-01-31
  - Seeding: 2009-10-01

Mark all plants with an individual plant identification number (ID) and note to which plot in the field they belong (later used to do tests for independence of effects from heterogeneities in the field).

Climate chamber: randomly rearrange pots every day.



# Chapter 5

## Measurement protocol

### 5.1 Equipment

- Ruler or measuring tape, 40 – 50 *cm* (glued to the edge of a lab table), measuring accuracy: 1 *mm*
- Scale (accuracy: 0.01 *g*)
- Digital camera (always take picture with a ruler beside the object!!)
- Labels (cardboard, with a wooly thread, to mark plants)
- Water- and UV-proof felt pen (or pencil)
- Paper notebook
- Set square, 14 *cm* diameter (to measure short lengths and angles)
- Calliper (accuracy: 0.1 *mm*)
- Sharp knife or scissors, or half razor blade (for cutting the stem)
- Leaf area meter (measures in *cm*<sup>2</sup>)
- Spade
- Bucket
- Plastic bags
- Drying oven
- Clean aluminium trays (different sizes: e. g. 20 \* 10 \* 10 *cm* or smaller)
- Photocopier (always with a ruler!!) (Prefer to use a photocopier to keep the scale of the object instead of a flatbed scanner.)

- LICOR or LCPro photosynthesis rate meter
- Temperature sensors, automatic weather station or Min/Max thermometers

## 5.2 Vegetative stages

### 5.2.1 Destructive measurements in the field and climate chamber:

1. Field: (on a not-too-wet day) Dig out three plants with a spade to recover the entire root system. Chamber: unpot three plants. Carefully remove the soil without removing the fine roots
2. Bring the plants into the lab, wash off the roots in a bucket with clean tap water
3. remove excess moisture from root system (kitchen roll) and place the cleaned plant onto a sheet of black plastic or paper and take pictures of the plant (from different positions, details of root system). Note the image number and the label of the plant.
4. Recover the roots that have fallen off into the bucket, put everything into a plastic bag and store the whole plant in a fridge or cooling chamber until further use (The plants must be processed quickly! Don't keep longer than one day!)
5. Determine stem length with ruler
6. From the base of the shoot, write rank on leaf blade, remove leaves and lay them out on the table from left to right. If you encounter a side shoot, place it on the table above the subtending leaf. Take a picture. Place defoliated stems back into the plastic bag.
7. Determine fresh weights of leaves
8. Determine length (blade plus petiole together), width and leaf blade area, using leaf area meter, then place all leaves on a scanner with a measuring stick (e.g. ruler) (label the photocopies). These copies will be our backup system!
9. Put each leaf in its own aluminium tray and place a label in it indicating rank and order.
10. Repeat steps 8 – 11 for the side shoots
11. For each side shoot cut the stem into its internodes with a scissor or knife and lay out the internodes from left to right.
12. Determine fresh weight of each internode
13. Determine length and diameter of each internode (alternatively, only lowermost and uppermost internode)
14. Place each internode in its own tray and label with rank, order and a running number indicating the number of the side shoot.
15. Repeat steps 13 – 16 for main stem.

16. Determine fresh weight of root system and place it into a tray with label of plant.
17. Dry everything for 48 hours in a drying oven at 105 degrees. (alternatively: dried 48 *h* at 70°C followed by 24 *h* at 105 °C)
18. Determine dry weights of all organs.

You can use the template on page [79](#) for your measurements.

### 5.2.2 Non-destructive measurements including scanning:

1. Label five plants per chamber (three temperature treatments)
2. Draw topological map of plant (indicating number of phytomers on main and side stem(s), position of side stems (rank of mother node on main stem)).
3. Take pictures of plant: The whole plant from four different sides
4. Count number of leaves
5. For main stem: Measure stem height
6. Length and width of leaves
7. Developmental stage of leaf (1: juvenile (still extending), 2: mature (extended), 3: senescent (to be defined)).
8. Phyllotactic and divergence angles of leaves and shoots
9. Phyllotactic and initial angles of side shoot and main shoot
10. Repeat steps 5 – 8 for all side shoots (if applicable)

You can use the template on page [82](#) for your measurements.

Further data needed for the model (template on page [78](#)):

- Daily control of phenology/developmental events of the labelled plants
- For each organ type (leaf, internode, leaf, bud, flower, fruit, ... ):
  - Count the number per stem / per leaf
  - When does a new organ appear?
  - When does it die off?
- number of lateral shoots (when does a bud break?)

### 5.2.3 Photosynthesis measurements

Using an LCPro or LiCOR photosynthesis meter determine light response curves (net photosynthesis rate from 1200 to 0 micromols PAR m<sup>-2</sup> s<sup>-1</sup>), stomatal conductance and leaf temperature at ambient  $CO_2$ , of three main stem leaves of five plants. See template at chapter [B.6](#).

### 5.2.4 Measurement of microclimate

Continuous measurement of local temperature with temp. sensors (an automatic weather station).

At least minimum and maximum temperature per day to calculate the average temperature.

## 5.3 Generative stages

Measurement plan for generative stages (from week 7)

Dynamics of inflorescence length (every 3 days). For all observed stages, you should note the first time you observe the stage and the time when 50

- Infl. bud appearance
- Flowering
- Flowers drop
- Fruits appearing
- Harvest maturity

### 5.3.1 Destructive measurements:

1. Intermediate harvest (at flowering and fruit growth):

- Harvest 10 inflorescences of main stems and side stems (1st order) each
- Measure length of main or side stem
- Length of inflorescence
- Basal rank of inflorescence (number of phytomers below the inflorescence base + 1)
- Length of internodes and peduncles (from base to top, indicating rank)
- Separate peduncles from flowers
- For all flowers note:
  - Order and rank



- Developmental stage (1: opened or 2: closed)
    - Measure length, width or diameter
    - FW and DW
  - For all peduncles: FW and DW of all peduncles of an inflorescence
  - For all fruits note:
    - Order and rank
    - Developmental stage (1: immature, 2: mature)
    - Measure length, width or diameter
    - FW and DW
2. Final harvest (harvest maturity):
- Harvest and measure 10 main and side stems (as above)
  - and additionally:
  - Number of aborted flowers (just a peduncle left)
  - Phyllotactic and divergence angles of peduncle and shoot
  - Number of fertilized fruits
  - From 2 main and 2 side stems, harvest fruits from bottom to top
  - Separate peduncles from fruits and measure peduncle length
  - Pool all peduncles and determine DW



# Chapter 6

## Time schedule

Example of a time schedule for the measurement protocol, at the example of measurements conducted oilseed rape seedling sown in winter in the greenhouse and in climate chambers at Zhejiang University, Hangzhou, P.R. China.

The daily resolution of the FSPM-P dictates the type and frequency of measurements needed. This means that the frequency of non-destructive measurements is ideally daily, but realistically three times per week.

Measuring intervals

destructive measurements:

plants are selected randomly

Climate chamber

Five plants per climate chamber per measurement date every 14 days

Field

Five plants per measurement date every 14 days

non-destructive measurements:

plants are labelled to identify them

Climate chamber, field, greenhouse

Five plants per climate chamber per measurement, three times per week (extension growth of leaves and internodes). Do not use the same plants every three days

Ten plants per climate chamber per measurement daily: number of leaves, developmental state

General remark: For organisational reasons, destructive and non-destructive measurements can

actually not be carried out on the same day unless you have unlimited human resources . . . .

Each block consists of five marked plants

After the end of January, all climate chamber plants are put into vernalisation (unheated greenhouse, but ideally 5 degrees centigrade).

regular check of flower initiation (with field plants, under the dissecting microscope): determine temperature sum of flower initiation.

# Chapter 7

## 2D Digital Measurements

This chapter describes how to do destructive leaf measurements by applying photogrammetric methods to 2D images.

Please consider if you really want to do this - doing it manually with a ruler and leaf area meter is mostly more efficient.

### 7.1 Image Capture

First you have to think about how to obtain your images. There are two main methods: using a digital camera, or using a scanner. Both methods have their pro's and con's:

	Photo camera	Scanning
<b>pro</b>	<ul style="list-style-type: none"><li>• flexible in use</li><li>• suitable in the field</li><li>• allowing non-destructive measurements</li><li>• suitable for online measurements (monitoring development)</li></ul>	<ul style="list-style-type: none"><li>• no problems with shadows or difficult light conditions</li><li>• resulting images are better (always clear)</li></ul>
<b>con</b>	<ul style="list-style-type: none"><li>• difficult light conditions (shadows)</li><li>• problems with focusing (risk of blur)</li></ul>	<ul style="list-style-type: none"><li>• not suitable for use in the field (requires a connection to a power socket and is not water or splash proof)</li><li>• some (mature) leaves might be too big for the scanner (e. g. in cucumber, tomato, rapseed)</li></ul>

The use of a scanner might thus be more advantageous when doing measurements on harvested

material, whereas a camera might be the tool of choice in the field and when monitoring dynamics of development.

### 7.1.1 Equipment

- digital camera or flat bed scanner with a minimum resolution of at least six mega pixels for the camera, respectively 200 dpi for the scanner (use uncompressed image formats)
- a reference of known size (a ruler or small post-it notes)
- only for the camera:
  - if you have leaves that are rigid and at the on top of the leaves (if possible: coated glass, reducing reflections)
  - blue mat material (e. g. velvet) as background (optional; blue gives the best contrast to the plant material as the latter hardly contains blue pixels)
  - if necessary use macro lenses
  - if indoors, additional halogen floodlights (or equivalent) will be needed (you cannot use a flash because of the reflections from the glass plate)
  - if possible, construct a portable scaffold of wood or metal and place the camera into a perspex case. Drilling a hole of the size of the camera objective into the bottom of the case will allow you to place the camera into it, facing down. Advantage: The distance between camera and plant object is the same, facilitating subsequent (semi)automatic image analysis.

If you use a very white background you will often get problematic small shadows around the leaf like in figure ??.

Another useful idea is to use rastered paper like graph paper or squared paper to have an additional yardstick. This may work but sometimes you will get problems with thresholding to separate the background from the leaf (in the foreground), see figure ??.

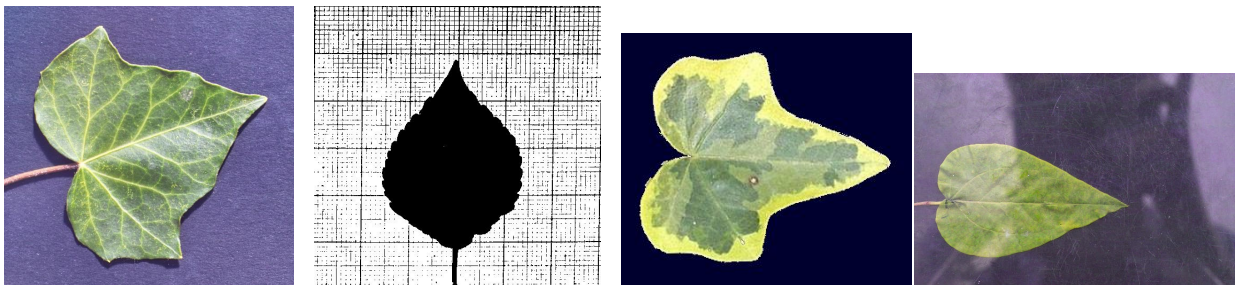


Figure 7.1: Examples of some problematic images

### 7.1.2 Measurement Protocol

Measurements carried out on images obtained with a digital camera or a scanner.

If you use a digital camera you additionally (and always!) have to care about:

- good light conditions, i.e. equally and well-lit (no shadows, no single (sun-)light rays, no differently lit areas). Best if light is diffuse.
- height contrast [Vaia: Do you mean background material??]
- camera focus (Most lenses have problems focusing at near distances, so you have to turn the auto focus off and do it manually. Sometimes, this problem also occurs with macro lenses.)
- try to minimize glare (which also means that you cannot use a flash light when you use a glass plate on top of the object to flatten it)
- zoom in (as much as possible) in order to exclude unnecessary background which will have to be removed from the image later anyway.
- However, if possible, you should set up a fixed scaffold, ensuring a fixed distance between camera and plant: in that case the zoom does not need to be changed.

Taking digital images of leaves:

1. For each leaf:
  - (a) Prepare a label or label the reference (if you use a post-it note)
  - (b) Spread the leaf out together with the reference
  - (c) Take care that leaf/leaves and reference do not cover each other
  - (d) Take the photograph or do the scan
  - (e) (Scanning only) save the image with a unique and significant name (including plant id, leaf rank and order, date, time)
  - (f) Note the name of the image and save it in your documents
2. Use an order for photographs (so that labels don't have to be used in the photographs). The following sequence is suggested:
  - (a) Leaves of main stem ordered by rank (from bottom to top)
  - (b) After this leaves of side stems ordered by rank (from bottom to top)

## 7.2 Leaf Image Analysis

There are many possible ways to do photogrammetric measurements. For our purposes we decided to use ImageJ software <sup>1</sup>

### 7.2.1 Surface area

The method described here is based on the ImageJ "Tutorials and Examples"<sup>2</sup> page, where you can find a documentation for "Area Measurements and Particle Counting"<sup>3</sup>. This example provides a step by step method to measure leaf area.

Measuring leaf area from an image is essentially easy. You just have to separate the background from the leaf by using a threshold value, then count the number of black pixels and calculate the corresponding area. However, always remember that the devil is in the details..

Platzhalter	Open leaf image via Select <b>File → Open...</b>
Platzhalter	Optional: <ul style="list-style-type: none"> <li>• Crop image: If your image includes problematic areas, such as edges of your photos, shadows, unwanted objects at the edges of the image, you can Crop your image by dragging the Rectangular tool then going to <b>Image → Crop</b></li> <li>• Sharpen: <b>Process → Sharpen</b></li> <li>• Enhance Contrast: <b>Process → Enhance Contrast</b></li> </ul>


<sup>1</sup>Digital Image Analysis Software – Download ImageJ 1.4.2 (current as of March 14, 2010); downloadable for free from: [11]

<sup>2</sup><http://rsbweb.nih.gov/ij/docs/examples/index.html>

<sup>3</sup><http://rsbweb.nih.gov/ij/docs/pdfs/examples.pdf>



<p>Platzhalter</p>	<ul style="list-style-type: none"> <li>• Convert color image to grayscale by: <ul style="list-style-type: none"> <li>– To get maximum contrast it is useful to split the image into RGB-color channels  <b>Image → Color → Split Channels</b>  and only use the green channel (or another one, depending on leaf colour)</li> </ul> </li> <li>or <ul style="list-style-type: none"> <li>– Convert scanned color image of leaf to grayscale:  <b>Image → Type → 8-bit</b></li> </ul> </li> <li>• Set measurement scale:  Draw a line over a 50 mm section of the ruler then <b>Analyze → Set Scale...</b>  In the Set Scale window enter 50 into the 'Known Distance' box and change the 'Unit of Measurement' box to mm, check 'Global'</li> <li>• Draw a new line and confirm that the measurement scale is correct.</li> </ul>
<p>Platzhalter</p>	<ul style="list-style-type: none"> <li>• Threshold new image of leaf using manual settings:  <b>Image → Adjust → Threshold</b>  and adjust the sliders until the entire leaf is included in red and click 'Apply'</li> </ul> <p>The manual threshold setting includes (hopefully) all of the leaf area.</p>
<p>Platzhalter</p>	<ul style="list-style-type: none"> <li>• Remove Noise (optional):  <b>Process → Noise → Despeckle</b>  Specks don't have to be deleted from the background, nor do holes in the images because they can be adjusted for with the image analysis. N.B. Was meinst Du damit??</li> <li>• Cut petiole:  <b>Image → Adjust → Threshold</b></li> <li>• Fill holes (optional):  <b>Process → Binary → Fill holes</b></li> </ul>

	<ul style="list-style-type: none"> <li>• Calculate area of entire portion: Surround the leaf with the rectangular selection tool <b>Analyze → Analyze Particles</b> Enter 50 as the minimum particle size, toggle 'Show Outlines', check "Display Results" and click 'OK'</li> </ul> <p>Outline of analysed area will be drawn. Data window gives an area in <math>mm^2</math>, depending on the calibration setting.</p>
---	---

### 7.2.2 Other measurements

If you have set up the measurement scale once, you can also measure the leaf length and width as well as the length and diameter of the petiole just by drawing a line.

### 7.2.3 Semiautomatic Measurements

If you have qualitatively good images you can try to do your measurement (semi) automatically by using the powerful macro language that is included in ImageJ. You can do this if you have taken all your photos at the same focal distance without changing your reference size.

A proper way to do this is to create a directory for all new images and to apply your macro to it. Then the steps could be something along the following lines:

For all files in directory A do:

1. Open Image
2. Set measuring unit
3. Crop
4. Split Channels
5. 8-Bit Grey-Level
6. Threshold [Vaia: if your chosen threshold is big enough, then filling holes should not be necessary – it is even wrong as it falsifies the leaf area computation]
7. Make Binary
8. Open
9. Close
10. Remove and Close Stalk
11. Analyse Image
12. Save measured Data (Area, length, width, ...)
13. Close Image

In another project we have realised this for time sequences of poplar leaves of different individuals. In a second loop over all individuals all included directories (each directory represents a single day of measurements) will be analysed and the extracted data are written to separate

output directories. It includes input dialogues, interactive threshold manipulation, automatic leaf rotation and shifting (to get normed and comparable data), automatic petiole cut, etc., summing up to a total of about 640 lines of code. For more information, please contact the first author<sup>4</sup>.

ImageJ also provides another method to evaluate multiple images at once, the so called "stack method" (for more information [?, 1, 22]).

---

<sup>4</sup><mailto:mhenke@uni-goettingen.de>



# Chapter 8

## Data processing

Before the measured data can be used as model parameters they need to be subject to some form of data processing, mainly statistical analysis. Using statistical analysis one derives, e. g. simple linear or nonlinear regression models from the raw data, which can then be used in the model to reproduce the distribution of a parameter or the relationship between two parameters (e. g. length and width of the leaf blade).

One important aspect of data processing is the accurate and safe storage of data in a clear and easily retrievable form.

To manage this challenge you need knowledge about basic mathematical techniques, like regression analysis and statistics as well as the knowledge about corresponding software like the free, statistical software package **R**<sup>1</sup>.

### 8.1 Data processing and storage

Your data may be stored in Excel worksheets. Make sure you use one directory per plant, then one subdirectory per measuring date (in this subdirectory you also store the digital images and leaf scan files). For each file please use a name that is composed of the plant name, the date and time of measurement, e. g. “Plant08150911231000.xls”. Within an Excel sheet you use one worksheet per phytomer, plus a general worksheet for the plant-level measurements.

Store the same data as printouts (original hand-written notes and photocopies of leaves as a stapled file).

Derived parameters: If you have established the measurements from the same objects, then you can derive the following coefficients:

- Specific leaf area (SLA) of a single leaf = leaf area/leaf DW [ $m^2/kg$ ]

---

<sup>1</sup>The R Project for Statistical Computing <http://www.r-project.org/>

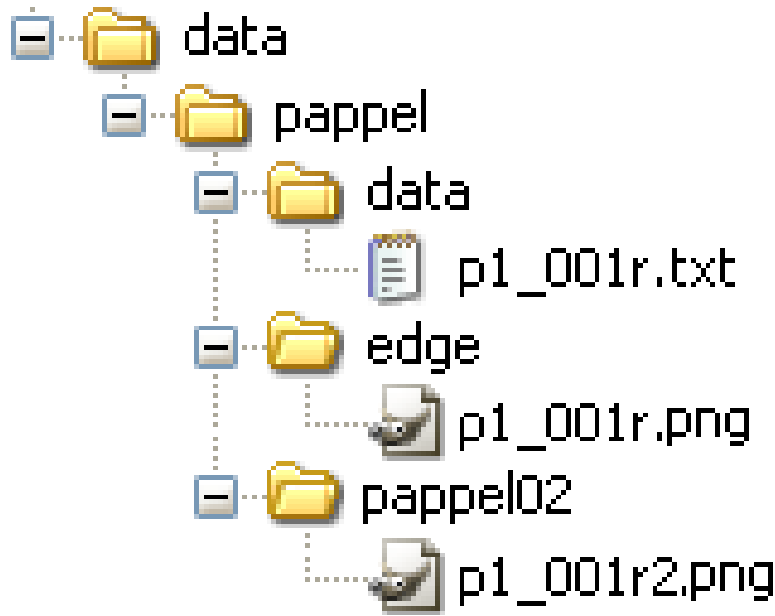


Figure 8.1: Example of a directory structure for data storage

- Specific leaf weight (SLW) of a single leaf =  $1/\text{SLA}$  [ $kg/m^2$ ]
- Specific internode length (SIL) =  $\text{len}/\text{DW}$  [ $m/kg$ ]
- Specific internode mass (SIM) =  $\text{DW}/\text{volume}$  [ $kg/m^3$ ]
- Number of flowers per inflorescence =  $\text{inflorescence mass}/\text{single flower mass}$  [-].

These parameters most likely are functions of organ age, rank and order.

## 8.2 Model specific

### 8.2.1 Conversion of dry weight to fresh weight

### 8.2.2 Average Dry weight

For  $ORGANS \in \{\text{Leaf}, \text{Internode}, \text{Peduncle}, \text{Flower}, \text{SeedandFruit}\}$ :

$$\text{AverageDWORGAN} = \text{DWofallORGANS}/\text{numberofORGANS}$$

### 8.2.3 Seed fertilization rate

$$\text{Seedfertilizationrate} = \text{numberoffertilizedseeds}/(\text{totalnumberofseeds}(\text{fertilizedandaborted}))$$

→

$$Fertilizationprobability = Seedfertilizationrate$$

N.B.: Hierzu braeuchte man eigentlich auch noch die genaue Verteilung (normal, Poisson, . . .).

#### 8.2.4 Microclimate / Environment

**Temperature** Use the daily recorded maximum and minimum temperatures to calculate the average temperature.

$$avgTempDay_i = (minTempDay_i + maxTempDay_i)/2 - baseTemp$$

**Radiation** Daily radiation reaching the ground is usually measured as shortwave (visible) plus longwave radiation. As a rule of thumb, about 50

**Humidity**

**Carbon dioxide concentration** Will usually be a constant for field crops, i.e. the ambient carbon dioxide concentration (383 ppm). If the crop is grown in a greenhouse, carbon dioxide concentration varies and its value then has to be read from an array of measured values.

**Wind**





# Chapter 9

## Results

In this chapter we present some of the results of our model. Several different developmental states of the simulated 3D-structure are shown, followed by a series of charts illustrating the dynamics of certain model parameters of the FSPM-P.

### 9.1 Morphology

Platzhalter

Figure 9.1: Seedling, X degree-days old.

### 9.2 Charts

...

Platzhalter

Figure 9.2: Mature flowering plant (age: X degree-days).

Platzhalter

Figure 9.3: Final stage of the FSPM-P after X degree-days.

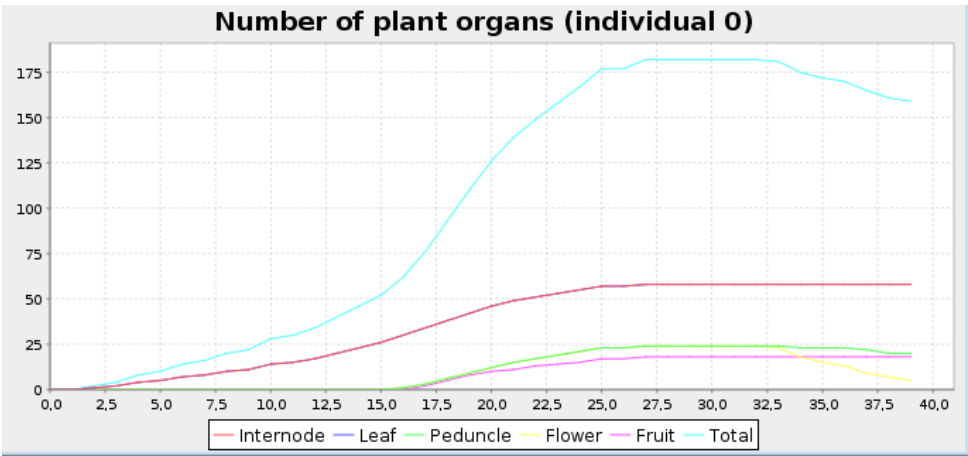


Figure 9.4: Dynamics of plant organ numbers during the simulated growth period.

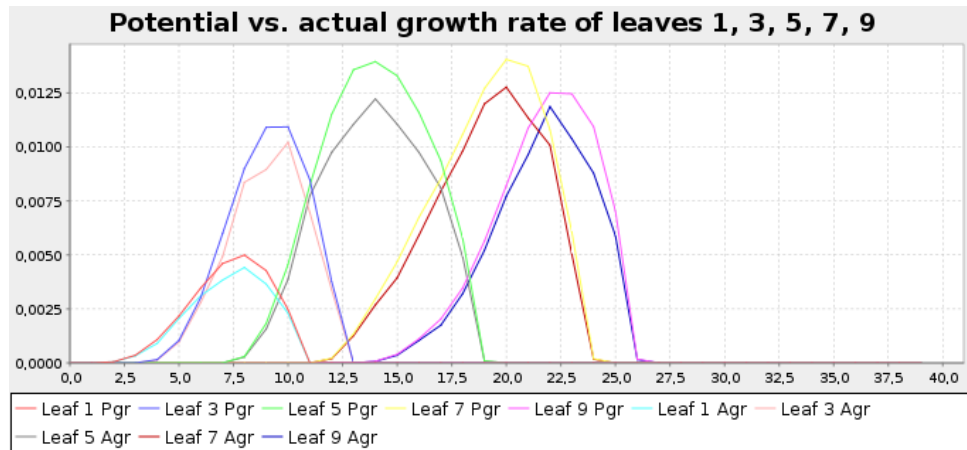


Figure 9.5: Potential vs. actual growth rate of leaves.

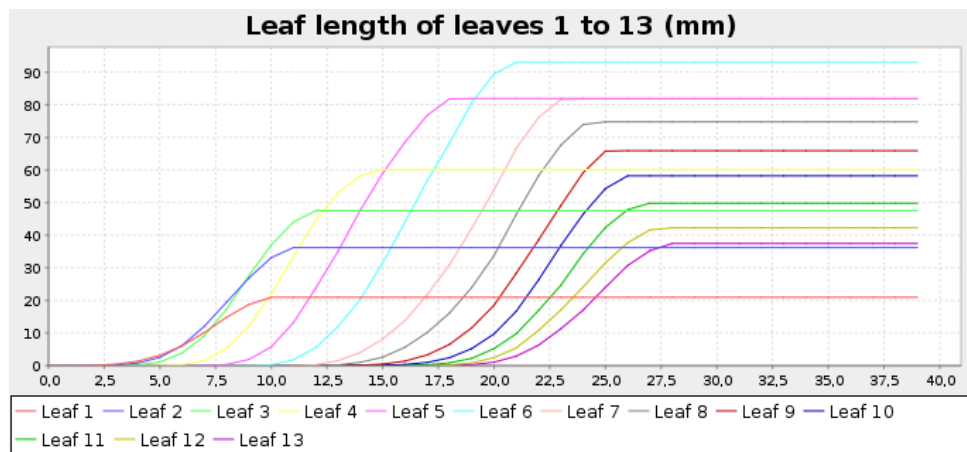


Figure 9.6: Dynamics of leaf length.

Platzhalter

Figure 9.7: Growth and maintenance respiration ( $g_C$ ).



# Appendix A

## Parameter List

This is a list of all parameters and methods the user has to change to get his own species-specific plant model. For a detailed description please read the chapter on model description. It can be used as a check list.

The lists are ordered by file name and with in each file by time of first occurrence and divided into parameters and methods.

### A.1 OrgansS1.rgg

#### A.1.1 Organ Parameters:

The following parameters are implemented for all organ types.

*ORGAN*  $\in \{ROOT, INTERNODE, BUD, LEAF, PEDUNCLE, FLOWER, FRUIT\}$   
**ORGAN\_ID, RATIO\_CARBON\_TO\_DRY\_WEIGHT\_ORGAN, MAIN\_ORGAN**  
**ASRQ\_ORGAN, MAX\_AGE\_ORGAN, FINISHED\_GROWING\_ORGAN**

*ORGAN*  $\in \{ROOT, BUD, PEDUNCLE, FLOWER, FRUIT\}$   
**FINAL\_DRY\_WEIGHT\_ORGAN**

**Q10MN, REFTMP, TEMPAIR, ORGAN\_MAT\_NUMBER**

#### A.1.2 Individual module:

Parameters:  
**SHOW\_INFO**

Methods:

*getCentralPool()*

### **A.1.3 Root module:**

Parameters:

Methods:

### **A.1.4 Bud module:**

Parameters:

Methods:

### **A.1.5 Internode module:**

Parameters:

Methods:

### **A.1.6 Leaf module:**

Parameters:

**SENESCENCE**

Methods:

### **A.1.7 Peduncle module:**

Parameters:

Methods:

### **A.1.8 Flower module:**

Parameters:

Methods:

### **A.1.9 Fruit module:**

Parameters:

Methods:

## **A.2 RulesS1.rgg**

### **A.2.1 Plant Parameters:**

## **A.3 Parameter.rgg**

Parameters:

**DEBUG\_MODE, RADIATION, POW, RANDOM, PHOTOSYNTHESIS\_MODEL**

Climate data:

**TEMPERATURE, TEMPERATURE\_SUM, TIME\_LENGTH, RELATIVE\_HUMIDITY,  
GLOBAL\_RADIATION, WINDSPEED, CLOUD\_COVER, CO2\_CONCENTRATION**

Methods:

## A.4 main.rgg

Parameters:

none

Methods:

*init()*, *grow()*

## A.5 Charts.rgg

Parameters:

INDI\_ID\_A, INDI\_ID\_B, CHARTS, CHARTS\_VS

## A.6 Others

### A.6.1 Textures

Usually, in order to obtain visually realistic simulations qualitatively good textures of all organs are required.



# Appendix B

## Templates

The templates provided here are consistent with the measurement protocol from chapter ?? . You can print them out and use them to make sure that you do not forget anything to measure.

Types of templates:

- Time schedule (Project plan)
- Time schedule (Measurements)
- Daily measurement
- Destructive measurements
  - Vegetative stages:
  - TODO Generative stage:
- Non-destructive measurements
  - Vegetative stages:
  - TODO Generative stage:
- Photosynthesis measurements

# B.1 Time Schedule (Project)

Main phases	Start	End
Preparation		
Seeding		
Data measurements		
Data processing		
Modelling		
Model calibration		

Additionally, you always have to count in at least 15 percent more time for documentation and paperwork at each phase.

## B.2 Time Schedule (Measurements)

### B.2.1 Vegetative stages

Start: \_\_\_\_\_ End: \_\_\_\_\_

#### Destructive measurements

Measurement interval: \_\_\_\_\_

Number of plants per interval: \_\_\_\_\_

Total number of plants: \_\_\_\_\_

Interval	Date	Plant IDs		
1				
2				
3				
4				
5				
6				
7				

#### Non-destructive measurements

Measurement interval: \_\_\_\_\_

Number of plants per interval: \_\_\_\_\_

Total number of plants: \_\_\_\_\_

Interval	Date	Plant IDs		
1				
2				
3				
4				
5				
6				
7				

B.2.2 Generative stages

Start: \_\_\_\_\_ End: \_\_\_\_\_

Destructive measurements:

Intermediate harvest

Measurement interval: \_\_\_\_\_

Number of plants per interval: \_\_\_\_\_

Total number of plants: \_\_\_\_\_

Interval	Date	Plant IDs		
1				
2				
3				
4				
5				
6				
7				

Final harvest

Date: \_\_\_\_\_

Number of plants: \_\_\_\_\_

Non-destructive measurements

Measurement interval: \_\_\_\_\_

Number of plants per interval: \_\_\_\_\_

Total number of plants: \_\_\_\_\_

Interval	Date	Plant IDs		
1				
2				
3				
4				
5				
6				
7				

### B.3 Daily measurements:

Daily control of phenology/developmental events of the labelled plants. (Use one template for each plant!)

Performing person:

Plant density:

**B.4 Destructive measurements (Vegetative):**

Date:	Time:	Operator:
Plant ID:	Plant location:	Plant density:
Climate:	Temperature:	Location of experimental site:

**General (steam/shoot) data:**

Order	Rank	Stem length	Number of leaves
1 (main stem)	——		
2	1		
2	2		
2	3		
2	4		

**Root system:**

Fresh weight:	Dry weight:
---------------	-------------

Leaf data:



## Internode data:

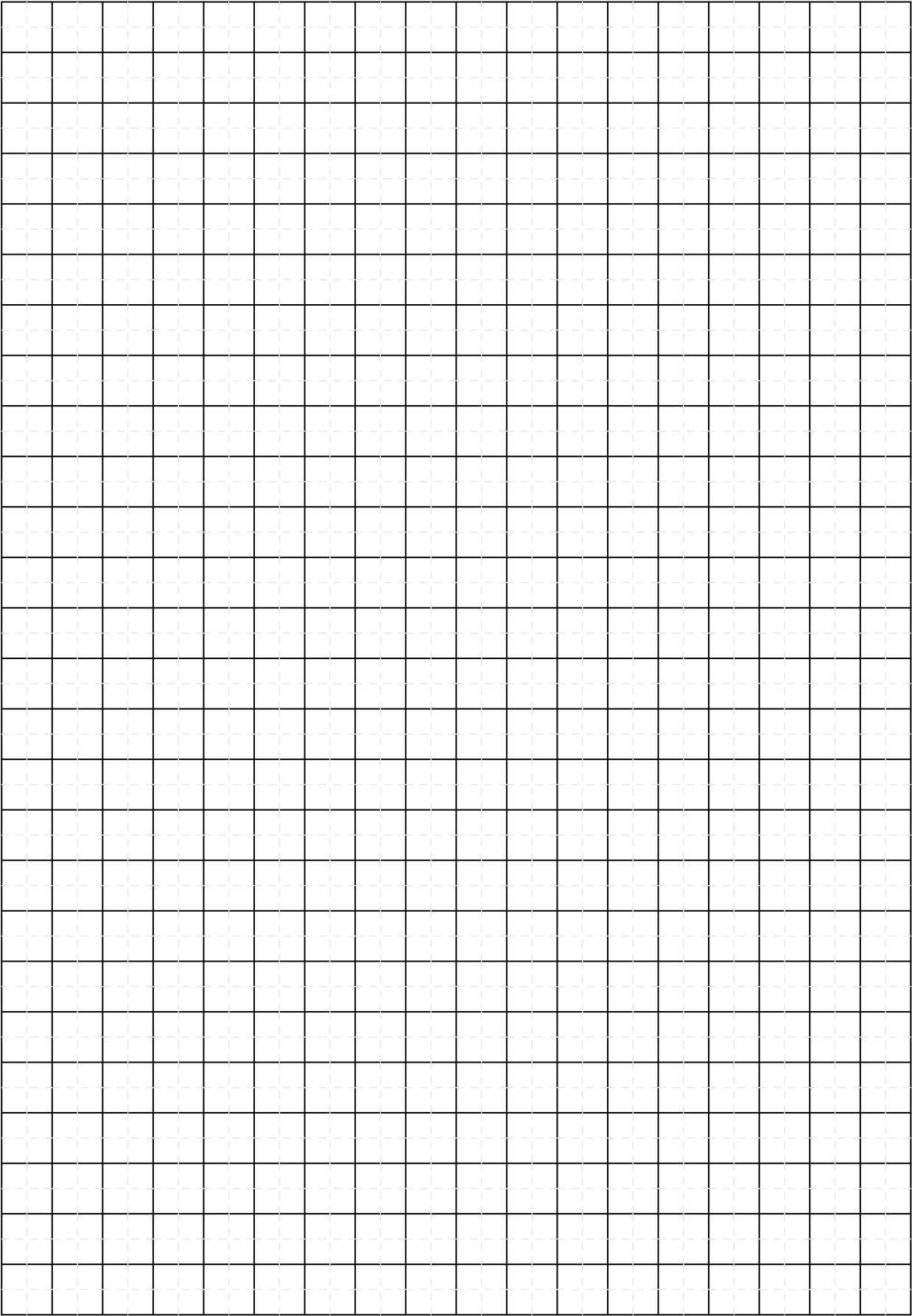
### B.5 Non-destructive measurements (Vegetative):

Date:	Time:	Operator:
Plant ID:	Plant location:	Plant density:
Climate:	Temperature:	Location of experimental site:

General (stem/shoot) data:

Leaf data:

Topological map:



# B.6    Photosynthesis measurements

Date:	Time:	Operator:
Number of plants: 5	Number of leaves per plant:	3 (on main shoot)
Climate:	Temperature:	Location of experimental site:

**Plant: 1**

Plant ID:	Plant location:	Plant density:
-----------	-----------------	----------------

no.	rank	total length	petiole length	blade length	width	developmental stage	photosynthesis ratio
1							
2							
3							

**Plant: 2**

Plant ID:	Plant location:	Plant density:
-----------	-----------------	----------------

no.	rank	total length	petiole length	blade length	width	developmental stage	photosynthesis ratio
1							
2							
3							

**Plant: 3**

Plant ID:	Plant location:	Plant density:
-----------	-----------------	----------------

no.	rank	total length	petiole length	blade length	width	developmental stage	photosynthesis ratio
1							
2							
3							

**Plant: 4**

Plant ID:	Plant location:	Plant density:
-----------	-----------------	----------------

no.	rank	total length	petiole length	blade length	width	developmental stage	photosynthesis ratio
1							
2							
3							

Plant: 5

Plant ID:

Plant location:

Plant density:

---

no.	rank	total length	petiole length	blade length	width	developmental stage	photosynthesis ratio
1							
2							
3							

# Appendix C

## Weather data

Weather or climate data are environmental recordings, which includes a detailed description of the test site. They are needed for the LEAFC3 and LEAFC3N photosynthesis model to work properly. They are defined as constants in the *Parameter.rgg* file (default) or can be loaded from climate file (*climate.txt*).

According to the model resolution the climate data resolution has also to be daily.

List of necessary data:

**TEMPERATURE** Temperature minus base temperature. [ $^{\circ}C$ ] (mean, daily)

**TIME\_LENGTH** Length of sun shine. [h] (total, daily)

**RELATIVE\_HUMIDITY** Relative humidity. [%/100] (mean, daily)

**GLOBAL\_RADIATION** Global radiation (long wave + short wave + diffuse). [ $W/m^2$ ] (mean, daily)

**WINDSPEED** Wind speed [ $m/s$ ] (mean, daily)

**CLOUD\_COVER** Average cloud condition, constant. (mean, per year)

**CO2\_CONCENTRATION** Ambient  $CO_2$  concentration [ $mol/mol$ ], constant. (mean, per year)

If you have daily data of cloud conditions or/and  $CO_2$  concentration, the environment file can be changed to a daily resolution quit quick.

The default weather data, used by model, are data from 2008 form Meteostation Haarweg - Wageningen, The Netherlands<sup>1</sup>, see figure C.1.

---

<sup>1</sup><http://www.met.wau.nl/index.html?http://www.met.wau.nl/haarwegdata/>

From file: <http://www.met.wau.nl/haarwegdata/dayfiles/2008/year2008hour.xls> converted to daily resolution.

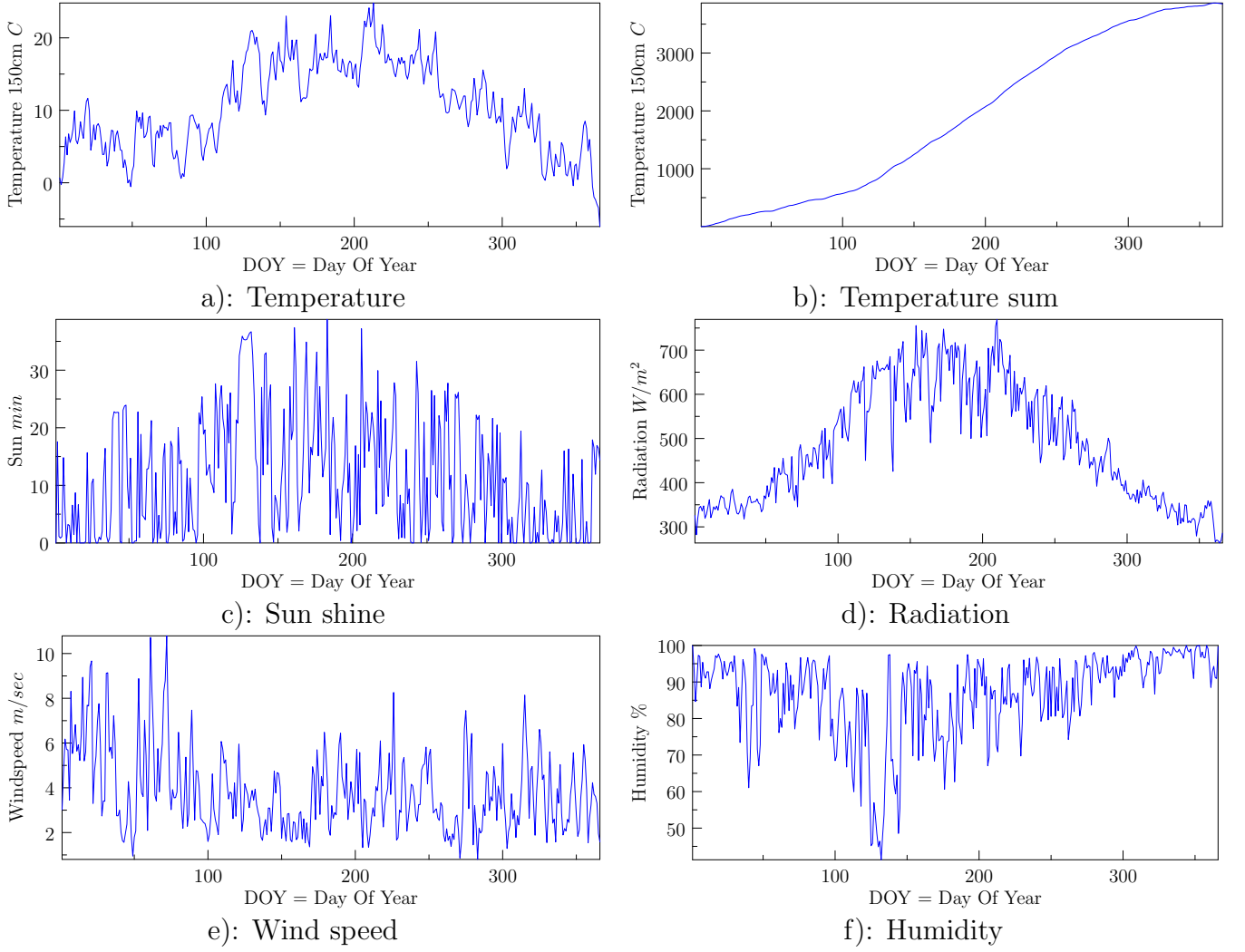


Figure C.1: Weather data form Meteostation Haarweg - Wageningen 2008

In the included climate file (*climate.txt*) of the FSPM-P are some more dataset prepared to compare plant growth on different climate conditions. The provided datasets are:

- Haarweg (The Netherlands)<sup>2</sup> 2007, 2006
- Göttingen (Germany)<sup>3</sup> 2009, 2008
- Los Baños (Indonesia)<sup>4</sup> 2009, 2008
- Hangzhou (China)<sup>5</sup> 2009, 2008

<sup>2</sup><http://www.met.wau.nl/index.html?http://www.met.wau.nl/haarwegdata/>

<sup>3</sup><http://www.ka.de>

<sup>4</sup><http://beta.irri.org/index.php/Climate-Unit/About-the-Unit/Weather-archive-Los-Banos.html>

<sup>5</sup><http://www.abc.de>



# Appendix D

## Textures

Uses textures of the model.



Figure D.1: Juvenile Leaf



Figure D.2: Adult Leaf

The senescent version of the leaf was produced by using an image manipulation program and modifying the colour values. However, normally, one should take scans of senescent leaves instead.



Figure D.3: Texture of an internode. This is also used for the peduncles.

# Appendix E

## Bugs and Problems

This is a list of know bugs and problems and some workarounds to solve them.

- Auslagern von Klimadaten in externe Dateien macht das Modell vielleicht langsamer??  
→ koennte daran liegen, dass die ganzen array dann keine constanten mehr sind und die Zugriffe somit langsamer sind.
- Viewing the simulated plant in OpenGL (Proteus) mode causes an Unexpected Exception:  
No OpenGL context current on this thread (probably not due to the model, but to the new OpenGL mode)



# Appendix F

## Version History

### **X, 2010 - scheduled**

**General FSPM 0.03** is released and available for download. The new features are:

- Extended and completed user manual:
  - Data processing part is written
  -
- The FSPM-Prototype has been extended and revised. New features:
  -
- A first paper is in preparation

### **June 27, 2010**

**General FSPM 0.02** is now available. The new features are:

- Revised and extended user manual:
  - Model description is extended
  - Extended time schedule
  - Glossary is completed
- The FSPM-Prototype has been extended and revised. New features:
  - Source-sink ratio as model regulation
  - Fixing some problems with the photosynthesis models

### **March 29, 2010**

**General FSPM 0.01** is now available. The new features are:

- A first draft version of the user manual is available.
- The FSPM-Prototype has been revised and some new features are added:
  - Restructuring the main file
  - Completing the organ class and extending the organ interface
  - Inserting a individual object with an identification number to model plant stands or to do inter-cropping
  - Three photosynthesis model are included (LEAFC3, LiethPasian, KimLieth)
  - Extend the chart class with new charts and the possibility to compare individuals
  - Climate data can be read from external files

**September 20, 2009**

**General FSPM 0.0001** - Project idea conceived. First version of the FSPM-Prototype implemented.

# Appendix G

## Symbols and units

This table contains all symbols used in the model, with names and units, in alphabetical order.

Symbol	Description	Unit of measure
PAR	Photosynthetically Active Radiation	$\mu\text{mol}_{\text{photons}}\text{m}^2/\text{s}$
PGR	Potential Growth Rate	$\text{g}/\text{d}$
AGR	Actual Growth Rate	$\text{g}/\text{d}$
LAI	Leaf Area Index	$\text{m}_{\text{leaf}}^2/\text{m}^2$
SLA	Specific Leaf Area	$\text{m}_{\text{leaf}}^2\text{kg}/\text{C}$
ID	Identification Number	—
SLM	Specific Leaf Mass	$\text{kg}/\text{m}^2$
SIL	Specific Internode Length	$\text{m}/\text{kg}$
SIM	Specific Internode Mass	$[\text{kg}/\text{m}^3]$

Species-specific input parameters (for LEAFC3 and KIMLIETH photosynthesis model):

Symbol	Description	Unit of measure
Vm25	Maximum carboxylation velocity at $25^\circ\text{C}$	$\mu\text{mol}/\text{m}^2\text{s}$
Jm25	Light-saturated potential rate of electron transport at $25^\circ\text{C}$	$\mu\text{mol}/\text{m}^2\text{s}$
Ej	Activation energy for electron transport	$\text{J}/\text{mol}$
Hj	Enthalpy parameter	$\text{J}/\text{mol}$
Sj	Enthalpy parameter	$\text{J}/\text{molK}$
theta	Coefficient controlling the smoothness of transition between light and temperature limitations on the potential rate of electron transport	—
Cdr	Proportionality coefficient for estimating leaf night respiration rate from Vm25	—
Kc25	Kinetic parameter for $\text{CO}_2$ at $25^\circ\text{C}$	$\text{mol}/\text{mol}$
Ko25	Kinetic parameter for $\text{O}_2$ at $25^\circ\text{C}$	$\text{mol}/\text{mol}$
f	PPFD loss factor	—

m	Composite stomatal sensitivity	—
Bs	Empirical constant for stomatal conductance	$mol/m^2s$
Dleaf	Leaf width (or needle diameter)	$m$
Aql	Leaf absorbtion coefficient for PPFD	—
Astom	false True = amphystomatous leaf, False = hypostoma- tous leaf	<i>boolean</i>

The following are the parameters for the SpeciesN definition according to the LEAFC3N photosynthesis model:

Symbol	Description	Unit of measure
Vm25Naref		$micromol/m^2s$
phi	photochemical quantum yield	$mole- / molquanta$
mNaref		—
Namin	minimum N content per $m^2$ leaf area	$g_N/m^2$
kVm		$1/g_N/m^2$
BsminNaref		$mol/m^2s$
k1gsmin		$1/g_N/m^2$
k0gsmin		—
k1m		$1/g_N/m^2$
k0m		—



# Appendix H

## Glossary

### H.1 Plant specific

**Phytomer (phyton)** A phytomer is a structural unit produced by a shoot apical meristem (SAM), continuously or rhythmically, throughout a plant's vegetative life-cycle. A typical phytomer consists of an internode, a node to which a leaf is attached, and lateral bud (containing another SAM) in the axil of the leaf. Variations to this general theme can be the lack of one of the mentioned organs or more than one axillary bud per phytomer, as in some tropical trees.

Initially, a young plant will produce a main stem due to the activity of the terminal SAM. At some later stage, first and higher-order branches are formed when some of the lateral buds break (i.e. start to grow out after a rest period).

**Meristem** Tissue in all plants consisting of undifferentiated cells (meristematic cells) and found in zones of the plant where growth can take place.

**Primordium** embryonic stage of an organ, after its initiation in the SAM

**Vegetative** (early) developmental phase of the plant in which leaves and stems are formed

**Generative** (later) developmental phase of the plant in which flowers and fruits are formed and after which the plant dies off if it is not perennial.

**Inflorescence** branched or unbranched structure on the plant which bears the flowers

**Senescence** last developmental stage of an organ characterized by processes such as cessation of, or decrease in photosynthesis, active reallocation of assimilates, and finally dehydration and decay.

## H.2 Data types

**Biomass** The dry or fresh matter of plants, produced by photosynthesis and growth.

**Biometry** a

**Metric** A trait is metric if it can be measured as a continuous size such as length, area, width, volume,...

**Meristic** A trait is meristic if it can be counted, e. g. the number of leaves per shoot, or the number of tillers (lateral shoots) in a cereal plant, are meristic traits

**Architecture** Architecture is the term referring to the topology (connection with each other) and geometry (arrangement in space), and size distribution (biometry and shape as a function of position) of plant organs. Plant architecture thus comprises the overall appearance of an individual plant as a result of organogenesis and morphogenesis

**Morphogenesis** Morphogenesis comprises all processes of organ extension in length and width, leading to the final dimensions of an organ. Growth processes in the form of biomass allocation to extending organs accompany these processes but are not strictly linked to them.

**Organogenesis** Organogenesis is the formation of new organs, as primordia in a SAM, due to controlled cell division and differentiation in certain tissue regions of the SAM.

**Phenology** Appearance of a plant or canopy during its development, usually defined by characteristic events, such as germination, appearance of the nth leaf, tillering/branching, flowering, fruiting, etc., as controlled by climatic conditions (temperature sum, day length, etc..).

**Photosynthetically Active Radiation** (PAR), the radiation usable for photosynthesis, roughly the visible spectrum of sunlight (400 - 700 nm).

**Photosynthesis** Photosynthesis is the active production of carbohydrates from  $CO_2$  and water in plant leaves using energy from sunlight (PAR).

**Environmental data** temperature, photosynthetically active radiation (PAR),

## H.3 Mathematical

**Fitting** The process of applying regression analysis to data. This method is sometimes called line-fitting or curve-fitting depending on the end result.

**Growth function** Any mathematical function that describes cell, organ, individual, or population growth, i.e. the dynamics of a state variable associated with growth (biomass, length, width, area) in time. Often, sigmoid functions, like the logistic, Gompertz or Richards curve are used.

**Regression analysis** Is a collection of statistical techniques for modelling and analysing several variables, describing the relationship between a dependent variable and one or more independent variables.

## H.4 Growth model

**Potential growth rate** maximum increment in length, area or dry weight of an (actively growing) organ per unit time under unlimiting conditions, i.e. if there is a surplus of assimilates and no shortage of nutrients, light and water. Usually described as the derivative of a logistic or beta function.

**Actual growth rate** actual increment in length, area or dry weight of an organ per unit time, usually observed under conditions of limited assimilate supply, shortage of nutrients, light and water, and/or competition/damage. According to some definitions, the actual growth rate may not be bigger than the potential growth rate, but that definition might be wrong under certain circumstances.

**Sink strength** Nutrients are transported via the phloem from sources (leaves and green internodes) to sinks (growing organs). Sink strength is defined as the rate of an organ to use these nutrients, e.g. for growth. A good proxy for sink strength is thus the instantaneous potential growth rate of an organ, but a mature organ can still exhibit a certain sink strength due to maintenance respiration.

**Maintenance respiration** amount of assimilates needed to maintain existing structural biomass, usually 1% of the structural biomass per day.

**Growth respiration** amount of assimilates needed for production of new structural biomass, usually an organ-specific constant depending on the composition of the newly formed biomass (proteins, carbohydrates, fats, minerals).

**Photosynthetically active radiation** PAR

**Leaf area index** (LAI), leaf area per square meter  $m_{leaf}^2/m^2$ ;  $LAI = C_{ha} * SLA$ , where  $C_{ha}$  is sum of carbon of all above-ground organs.

**Specific leaf area** ratio between leaf area and dry mass  $m_{leaf}^2 g/C$ . Its value is the leaf area [ $m^2$ ] that weighs exactly one gram; used as a conversion coefficient to compute leaf area from leaf dry mass.

**Specific leaf weight** (SLW), of a single leaf =  $1/SLA$  [ $kg/m^2$ ]

**Specific internode length** (SIL),  $SIL = len/DW$  [ $m/kg$ ]

**Specific internode mass** (SIM),  $SIM = DW/volume$  [ $kg/m^3$ ]

**Plastochron** refers to the rate of initiation of undifferentiated primordia.

The plastochron index and the leaf plastochron index are ways of measuring the age of a plant dependent on morphological traits rather than chronological.

**Phyllochron** time between emergence of successive leaves. = interval (time in degree-days between successive leaf tip appearance)

# Bibliography

- [1] E.M. Bakr. A new software for measuring leaf area, and area damaged by tetranychus urticae koch. JEN, 129(3):173–175, 2005.
- [2] Gerhard Buck-Sorlin, Benno Burema, J. B. Evers, G. van der Heijden, E. Heuvelink, L. Marcelis, P.C. Struik, P. de Visser, T. Damen, and J. Vos. Virtual rose: a new tool to optimize plant architecture in glasshouse rose production systems. Proceedings of the 5th International Workshop on Functional-Structural Plant Models, Abstracts of Papers and Posters, Napier (NZ), page 48, Nov. 4-9, 2007.
- [3] Gerhard Buck-Sorlin, Reinhard Hemmerling, Ole Kniemeyer, Benno Burema, and Winfried Kurth. A rule-based model of barley morphogenesis, with special respect to shading and gibberellic acid signal transduction. Annals of Botany, 101(8):1109–1123, 2008.
- [4] Gerhard Buck-Sorlin, Ole Kniemeyer, and Winfried Kurth. Barley morphology, genetics and hormonal regulation of internode elongation modelled by a relational growth grammar. New Phytologist, 166(3):859–867, 2005.
- [5] Gerhard Buck-Sorlin, Ole Kniemeyer, and Winfried Kurth. A grammar-based model of barley including virtual breeding, genetic control and a hormonal metabolic network, volume 3 of Proceedings of a workshop held in Wageningen (NL). Springer, 2007.
- [6] Gerhard H. Buck-Sorlin, Ole Kniemeyer, and Winfried Kurth. A model of poplar (populus sp.) physiology and morphology based on relational growth grammars. 2:313–322, 2008.
- [7] J. Goudriaan and H.H. van Laar. Modelling Potential Crop Growth Processes. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.
- [8] C. Groer. Dynamisches 3d-modell der rapspflanze (brassica napus l.) zur bestimmung optimaler ertragskomponenten bei unterschiedlicher stickstoffdüngung. Master’s thesis, University of Technology at Cottbus, Institut für Informatik, Informations- und Medientechnik, 2006.
- [9] C. Groer, Ole Kniemeyer, Reinhard Hemmerling, Winfried Kurth, H. Becker, and Gerhard Buck-Sorlin. A dynamic 3d model of rape (brassica napus l.) computing yield components under variable nitrogen fertilization regimes. Proceedings of the 5th International Workshop on Functional-Structural Plant Models, Abstracts of Papers and Posters, Napier (NZ), pages 4.1–4.3, Nov. 4-9, 2007.

- [10] Reinhard Hemmerling, Ole Kniemeyer, Dirk Lanwert, Winfried Kurth, and Gerhard Buck-Sorlin. The rule-based language xl and the modelling environment groimp illustrated with simulated tree competition. Functional Plant Biology, 35:739–750, 2008.
- [11] ImageJ. Digital Image Analysis Software - ImageJ. <http://rsbweb.nih.gov/ij/>.
- [12] Soo-Hyung Kim and J. Heinrich Lieth. A coupled model of photosynthesis, stomatal conductance and transpiration for a rose leaf (rosa hybrida l.). Annals of Botany, 91:771–781, 2003.
- [13] Ole Kniemeyer. Rule-based modelling with the XL/GroIMP software. In GWAL-6, pages 56–65, 2004.
- [14] Ole Kniemeyer. Design and Implementation of a Graph Grammar Based Language for Functional-Structural Plant Modelling. PhD thesis, BTU Cottbus, 2007.
- [15] Ole Kniemeyer, Gerhard Buck-Sorlin, and Winfried Kurth. A graph-grammar approach to artificial life. Artificial Life, 10:413–431, 2004.
- [16] Ole Kniemeyer, Gerhard Buck-Sorlin, and Winfried Kurth. GroIMP as a platform for functional-structural modelling of plants. pages 43–52, 2006.
- [17] Ole Kniemeyer, Reinhard Hemmerling, and Winfried Kurth. GroIMP. <http://www.grogra.de>.
- [18] Winfried Kurth, Ole Kniemeyer, and Gerhard Buck-Sorlin. Relational growth grammars – a graph rewriting approach to dynamical systems with a dynamical structure. In UPP, pages 56–72, 2004.
- [19] J.H. Lieth and C.C. Pasian. A model for net photosynthesis of rose leaves as a function of photosynthetically active radiation, leaf temperature, and leaf age. J. AMER. SOC. HORT. SCI., 115(3):486–491, 1990.
- [20] J. Müller, P. Wernecke, and W. Diepenbrock. Leafc3-n: A nitrogen-sensitive extension of the co2 and h2o gas exchange model leafc3 parameterised and tested for winter wheat (triticum aestivum l.). Ecological Modelling, 183:183–210, 2005.
- [21] N.T. Nikolov, W.J. Massman, and A.W. Schoettle. Coupling biochemical and biophysical processes at the leaf level: an equilibrium photosynthesis model for leaves of c3 plants. Ecol. Modelling, 80:205–235, 1995.
- [22] M.E O’Neal, D.A. Landis, and R. Isaacs. An inexpensive, accurate method for measuring leaf area and defoliation through digital image analysis. J. Econ. Entomol., 95(6):1190–1194, 2002.
- [23] J. H. M. Thornley. A model to describe the partitioning of photosynthate during vegetative plant growth. Annals of Botany, 33:419–430, 1969.
- [24] E. Veach. Robust Monte Carlo Methods for Light Transport Simulation. PhD thesis, Stanford University, 1998.
- [25] Lifeng Xu, Michael Henke, Jun Zhu, Winfried Kurth, and Gerhard Buck-Sorlin. Integrating quantitative genetic information in a functional-structural model of rice. page ?, 2010.

- [26] Lifeng Xu, Michael Henke, Jun Zhu, Winfried Kurth, and Gerhard Buck-Sorlin. A rule-based functional-structural model of rice considering source and sink functions. Proceedings of PMA2009: The Third International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications Beijing (CHINA), page 10, Nov. 09-13, 2009.
- [27] Xin You Yin, Jan Goudriaan, Egber A. Lantinga, Jan Vos, and Huub J. Spiertz. A flexible sigmoid function of determinate growth. Annals of Botany, 91:361–371, 2003.