Pasture Structure And Biomass Dynamic Analyzing System.

Haizhou Li

PASTURE STRUCTURE AND BIOMASS DYNAMIC ANALYZING SYSTEM

Author	LI Haizhou
Supervisor	Raphaël Martin
Associate supervisor	Wang Zhongjie
Internship organization	INRA
Filière	Internationale
Date of Defence：	September, 2011

哈爾濱工業大學

Université Blaise Pascal

Abstract

This thesis stems from research at UREP (Grassland Ecosystem Research Unit) at INRA (National Institute of Agronomic Research) in France. The aim of this thesis is to set up a field-scale system which investigates and analyzes the spatial structure and dynamics of managed, permanent grassland. Biologists will use this system to monitor the biomass and traits (main characteristics for species) of the pasture, to simulate agricultural management activity, to determine efficient agriculture strategy and to predict the future trends of the pasture under changing climatic conditions.

In order to carry out spatially-explicit analysis, the present vegetation growth model deals with a large plot of grassland and divides it into many small cells. Each cell has the same area (0.1 m^2; considered to be a bit-size area of vegetation for a domestic herbivore) but not necessarily same traits. These traits are grouped into patches, which are the combination of many common species in a same cell. Four common patches are described into the literature and are used to describe the grassland plot. Each cell is divided in four compartments depending on their developmental stage: green vegetative, green reproductive, dead vegetative and dead reproductive. This model not only considers the permanent structure of the pasture but also the agriculture activities which are simulated as three types of events: cut, fertilization, and grazing by animals. These events occur according to the management described by the biologist.

To maintain and utilize this model, a graphic user interface which is programmed in JAVA is implemented. This GUI interface is to specify the input of users and parameters from files. Meanwhile, it provides the methods to set the structure of the pasture. Finally, the output data is produced into the format CSV or NetCDF and displayed with a graphic chart. In the future, this model will be integrated into CAPSIS (Computer-Aided Projection of Strategies In Silviculture) platform which simulates the forest growth and presents a better result display (3D distribution graph chart).

Keywords: grassland, vegetation growth model, graphic user interface, CAPSIS

Content

Chapter 1 Introduction

1.1 Background and purpose of project

This project comes from UREP (Grassland Ecosystem Research Unit) which belongs to INRA (National Institute of Agronomic Research) in Clermont-Ferrand, France.

INRA is ranked number one agriculture institute in Europe and number two in the world. INRA carries out mission-oriented research for high-quality and healthy foods, competitive and sustainable agriculture and a preserved and valorized environment.

The Grassland Ecosystem Research Unit (UREP, UR874) is an INRA research unit attached to the Forest, Grassland and Aquatic Ecology Department (EFPA). The research of the UREP addresses the ecology, functioning and ecosystem services provided by permanent grasslands in a context of global change. This topical research area meets both theoretical (front line science) and applied, societal needs. The UREP project uses a pluridisciplinary approach at local and national levels via French (ANR) and European projects. It builds on experimental installations in situ (SOERE-ACBB, phenotyping platform) and on UREP expertise in functional ecology and the analysis of greenhouse gas balances.

The purpose of this project is to investigate seasonal and annual interactions between management and grassland dynamics. So a mechanistic system of the dynamics of production, and spatial structure in permanent pastures was constructed to fulfill the demand. The system is designed to respond to various defoliation regimes, perform multiple-year simulations and produce simple outputs that are easy to treat in order to extract spatial statistics. Some events (grazing, fertilization or cut) can occur during the simulation and allow simulating heterogeneity onto the grassland and biomass produced is computed at a daily time step. Output data is produced in different formats in order to carry out statistics, graphical representation, etc.

1.2 The status of related research

1.2.1 Research of UREP

The research theme of UREP is grassland ecology in response to global change. UREP uses a multidisciplinary research approach combining knowledge on grassland ecosystem structure, function and dynamics. UREP's three main areas of research in the last research period (January 2008-June 2010) were:

- Carbon and nitrogen cycling in grasslands and their consequences for the greenhouse gas balance;
- Linking grassland diversity and ecosystem processes;
- Grassland community dynamics under global change.

Significant advances were made in all three research areas over the last research period. These include improved understanding of the microbial processes controlling soil organic matter dynamics, the role of plant and soil community diversity in soil carbon sequestration and greenhouse gas emissions, the importance of grasslands as carbon sinks in Europe the links between above- and below-ground plant traits, the identification of plant strategies for the dominant grassland species. Phenotypic plasticity in aboveground traits has been demonstrated in response to defoliation and nitrogen addition. Other works on extreme climate events in grasslands suggests that forbs have a greater resistance to drought compared with grasses or legumes. Findings over the research period have also led to advances in the understanding of

- Community assembly rules for grassland species,
- Community structure (species/ functional traits) of grasslands in response to environmental and management changes, and
- Consequences of community structure for grassland production and feed quality. In grazed grasslands, the importance of herbivores and foraging behavior for the creation of short or high vegetation patches has been established.

For the next research period (2011-2014), the goal of UREP still is the same: contribute to sustainable ecosystem management in a changing environment. In order to achieve this, UREP is developing an integrated and predictive approach combining observation, experiments and modeling. The current research consists of two themed sections, linked by three cross initiatives.

Section 1 "C/N cycles & Greenhouse gases" focuses on the study of carbon and nitrogen cycling in grasslands in a context of climate change and sustainable agricultural practices. The two main objectives are

- Quantify C sequestration, N losses and emission of greenhouse gases from grasslands
- Identify the key mechanisms underlying C/N cycling in grasslands. This section comprises four subsections according to the scale of the study, ranging from the molecular level to the regional scale.

Section 2 "Biodiversity, dynamics and grassland function" focuses on functional ecology and community ecology. It consists of four subsections examining functional strategies of organisms and community structure, along with the role of plant-animal and plant-soil interactions. A key objective is to identify response traits of species and communities to management and climate factors, and to understand the role of these traits foe ecosystem processes.

The cross initiatives are designed to encourage interactions between the two themed sections, and to promote research synergies. These initiatives are:

- Strategic policy: progress in frontline science and scientific excellence;
- Development of innovative methods and shared research tools;
- Information transfer to stakeholders and farmers.

In conclusion, the current project for UREP over the next four years aims to develop widely-recognized scientific expertise and research specificity which meets the needs of sustainable agro-ecosystems under global change."

1.2.2 Management and Grassland pastures

Until the 1980s, French and European agricultural policies, by supporting certain products, encouraged intensification. The elimination of limiting growth factors resulted in a standardization of production systems, which also facilitated management practices. Furthermore, this is to field abandonment, depending on access or labor constraints (distance, slope, etc.) (Bignal and McCracken, 2000; Tasser and Tappeiner, 2002) [3]. Since the 1990s, agricultural policies have evolved and encouraged extensification by subsidies per hectare, such as a premium for grassland on livestock farms which limit their stocking rate in order to maintain or increase the biological diversity, which has become an important issue (Kirchmann and Thorvaldsson, 2000). In grassland zones dominated by natural pastures, an

important component of this biological diversity is attributable to the diversity of grassland species. This evolution is particularly relevant in mountainous zones where the areas used for farming are made up almost exclusively of natural grasslands (Flamant et al., 1999) [5]. In these situations, grazing or cutting operations make it possible to feed domestic herbivores, but are also a means of preventing woody species from colonizing these environments (Landsberg et al., 2003; Cole´no et al., 2005). Diversity of grassland species is generally assessed on the within-field scale (Grime et al., 1988; Bakker et al., 2004), or between farms and across regions (Thenail and Baudry, 2004) [7]. To allocate a field for a given use, farmers coordinate their decisions at the farm level (Papy, 1999; Cole´no and Duru, 2005). Thus, evaluating and predicting the impact of such policies on the diversity of grassland vegetation necessitate considering the farm level and assessing this diversity on the between-field scale. These policies led to the emergence of new areas of research and need the renewal of the reference data established for homogenous systems (van Keulen, 2006). In these complex systems, the farmer has to manage the diversity of his farmland. In this context, field characteristics have an effect on the distribution of practices. Studies showed the influence of distance to the cowshed, the surface area or the slope on the distribution of cutting and grazing practices (Morlon and Benoit, 1990; Josien et al., 1994; Fleury et al., 1996; Mottet et al., 2006). But field characteristics can also play a role on grassland vegetation diversity. The altitude or the aspect of the field can potentially influence plant species (Bornard et al., 2004) leading to differences in growing conditions (temperature and radiation) for a particular field (Legros et al., 1997). Farmers can themselves influence grassland vegetation by choosing management practices relevant for production or environmental purposes (Ronchi and Nardone, 2003; Cole´no et al., 2005). However, these studies are partial; none of them taking into account the main characteristics that play a role both in land use management practices and in the plant species found in natural grasslands. In other words, they do not really assess the complexity of the relationships involved in these livestock systems [1]

Numerous studies have shown that the chemical composition of grass is influenced by the season of growth and age of regrowth (Wilman et al., 1976; Demarquilly and Andrieu, 1988), as well as by the time of day (Holt and Hilst, 1969; Lechtenberg et al., 1972). Under strip-grazing management, defoliation of the sward

takes place by successive layers from the top of the canopy (Wade, 1991), so the chemical composition of the selected grass is a function of the vertical distribution of the chemical constituents in the sward. In order to predict the chemical composition of the ingested herbage, it is necessary to know this vertical distribution and the main factors controlling its variation. In grass species, it is well established that the bulk density of the sward increases with the depth of the sward, as well as the proportion of sheaths, stems and dead tissue (Wilkinson et al., 1970; Clark et al., 1974) [10]. The leaf blades are generally more digestible, richer in crude protein and poorer in cell-wall constituents than sheaths and stems (Deinum and Dirven, 1975; Wilman et al., 1976). There is, thus, an increasing or a decreasing vertical gradient of composition according to the chemical constituent. Such a vertical gradient in composition has been described in tropical grasses (Wilkinson et al., 1970; Stobbs, 1975; Hendricksen and Minson, 1980; Herrera et al., 1984) and on grass/legume mixtures (Clark et al., 1974; Holmes et al., 1992; Johnston et al., 1993; Wilkins et al., 1995). Very few data are available on pure temperate grasses under rotational grazing. In particular, there are few descriptions in the literature concerning the vertical gradient of chemical constituents in swards of perennial ryegrass (Lolium perenne L.), which is certainly the most widely used grass species for grazing in temperate environments. Similarly, the factors controlling the vertical variation of chemical composition in swards have often been the subject of separate studies (Wilkinson et al., 1970; Herrera et al., 1984; Johnston et al., 1993). The aim of the present study is to describe the changes in sward structure (biomass, chemical composition) of a perennial ryegrass sward over the growth season for grasslands subjected to different management and climatic conditions.

1.2.3 Dynamic modeling of pastures

Dynamic, mechanistic and deterministic models can greatly assist ecological investigation. Dynamic models of managed grasslands already exist. They usually predict daily growth as the product of potential growth and a number of functions of environmental factors (temperature, water, nutrient supply and season). In the earliest models, potential growth was determined empirically from the plant's genetic potential or from field measurements. More recent models have used a mechanistic approach, based on light-utilization efficiency, which enables a better understanding of seasonal dynamics of production. Several models predict the

quality of the forage by estimating either the proportion of green and dead material or the digestibility of herbage, often in relation to the proportion of green leaves, stem and senescent material. [4] The existing models are based on single species or growth forms, and the models that predict the behavior of diverse pastures consider each species or growth form separately, which brings in a great complexity in inputs and outputs for highly diverse pastures. The objective here was to develop a spatially-explicit model of permanent pasture, capable of simulating the effects of management (type and intensity) on biomass and sward structure at the field scale. The model had to be as simple as possible and had to produce outputs that could be used directly as inputs for spatial statistics. To keep the model simple, a functional approach rather than a species-based approach was chosen; it was assumed that the dynamics of permanent pastures could be explained by the average biological attributes (functional traits) of the plant community making up the grassland. The present model builds on a simpler, pre-existing vegetation model described in the papers of Jouven et al (2006) [1].

1.3 Main content and organization of the thesis

This thesis implements an analytical system to investigate and analyze the spatial structure and the biomass of a permanent pasture dynamically. The work is divided into two parts: an analytical model which simulates the spatial structure and the flows of biomass and a graphical user interface to run and use easily by providing means to choose input files and generate output data in a correct format.

The organization of the thesis is as following.

Chapter 1 describes the background of this system, its purpose and the related research in its field.

Chapter 2 presents the vegetation growth model and relations between variables in this model.

Chapter 3 indicates the system requirement analysis including general system requirements, the functional and un-functional requirements.

Chapter 4 specializes the system design which is separated into several parts: system structure design, IO part specific design, GUI module specific design and Logic module specific design.

Chapter 5 narrates system implementation and testing including the environment of system implementation, implementation of modules, key interface of the software system and system testing.

Chapter 2 Vegetation Growth Model

The aim of this system is to investigate and analyze the dynamic of grassland biomass production and its spatial structure. It simulates management events applied to the grassland and computes flows of biomass inside the pasture. This system consists of a model which describes methods to analysis the structure and the biomass of the pasture and a graphic user interface which provides methods of determining the structure of the pasture. Meanwhile, this user interface maintains and runs the simulation of the model. In this chapter, the vegetation growth model is presented in terms of computer science.

2.1 Model description

The original vegetation growth model is based on biological principles. To implement the model integrated into the software, the biological description needs to be translated into the language of computer science. This is achieved using the process description tool “Berkeley Madonna”. In addition, “Berkeley Madonna” can set the value of variables and display the result of these variables through the process which has been set up. Thanks to “Berkeley Madonna”, this vegetation growth model has been described in computer science terms.

Given that this model strongly relates to the biology and agriculture, there are many specialized words and abbreviations. Table 2-1 shows all the abbreviations needed in this thesis.

Table 2-1 Abbreviations explanation

Abbreviations	Explanation
ABS	abscission
AET	actual evapotranspiration
AGE	age expressed in units of thermal time
BM	biomass
DR	dead reproductive
DV	dead vegetative
ENV	environmental variables related to soil and climate characteristics
GV	green vegetative
GR	green reproductive
K_VS	coefficient of vegetative senescence
KI_VS	coefficient of vegetative abscission
K_RS	coefficient of reproductive senescence
KI_RS	coefficient of reproductive abscission
LAI	leaf area index
LLS	leaf lifespan
NI	nutrition index
OMD	organic matter digestibility
PAR	photosynthetically active radiation
PET	potential evapotranspiration
Percent LAM	percentage of laminae in GV
PLUVIO	Precipitation
REP	reproductive function
RUEmax	max radiation use efficiency
SEN	senescence
SEA	reserve storage and mobilization
SLA	specific leaf area
ST	sum of temperatures
W	water stress
WR	ratio of water reserves
WHC	soil water-holding capacity

This model simulates the grassland at the field-scale and takes into account its spatial heterogeneity. To simplify the situation and facilitate the model development, the hypothesis of modeling is to divide the field into many small cells (0.1 m^2), assumed to be the equivalent of a bite-size area of vegetation for domestic herbivores. In each cell, vegetation is characterized by a series of traits. This set of traits is named a patch. Adjacent cells with the same patch type are grouped as a site for practical computational purposes. Site is a kind of virtual unit which contains one or many cells of the same type. To resume, site is always analyzed as a whole part a field comprises of many sites. Chart 2-1 illustrates the logical structure of the grassland including different cells and sites across the field.

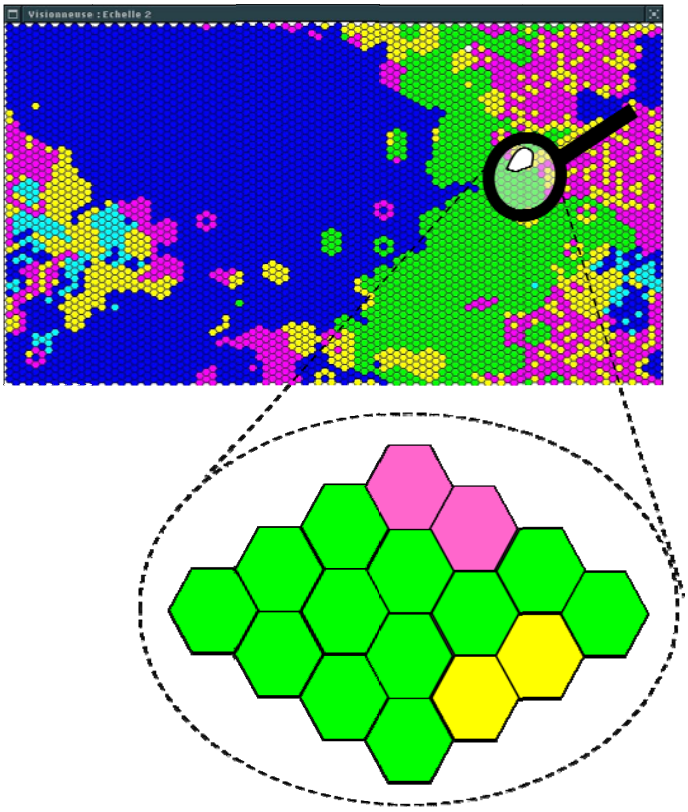


Chart 2-1 Logical structure of the grassland

Grassland communities are described using a set of average functional traits of their constituent grass groups. The sward is subdivided into four structural compartments: green leaves and sheath (GV or green vegetative), dead leaves and sheath (DV or dead vegetative), green stems and flowers (GR or green reproductive), and dead stems and flowers (DR or dead reproductive). Each compartment is characterized by its biomass and age. Only above-ground growth is modeled, using a light-utilization efficiency approach modulated by a seasonal pattern of storage

and mobilization of reserves. Age affects senescence, abscission and digestibility of green compartments and, therefore, the quality of green leaves and stems can increase or decrease over time in relation to net growth and defoliation dynamics. The functional traits having the greatest impact on model outputs are seasonal effects, period of reproductive growth and effects of temperature on photosynthetic efficiency. The functional traits of the grass groups were parameterized for temperate pastures of the Auvergne region in France. The other model inputs are few: proportion of functional groups, basic weather data (incident photosynthetically active radiation, mean daily temperature, precipitation and potential evapotranspiration) and site characteristics (nitrogen nutrition index, soil water-holding capacity). The system can be applied at a field scale.

The basis of the model is a vegetation growth model describing the natural life cycle of the pasture without considering agricultural management (for example: cut, fertilization, etc…). Vegetation growth and development occurs each day and we update the situation of the pasture on a daily timescale. To simulate the life cycle of the grassland, growth, senescence and abscission are described as continuous flows. So the pasture can be classified into 4 compartments within the whole life cycle of the grassland. Chart 2-2 shows the biomass flow through these 4 compartments within the life cycle of the grassland.

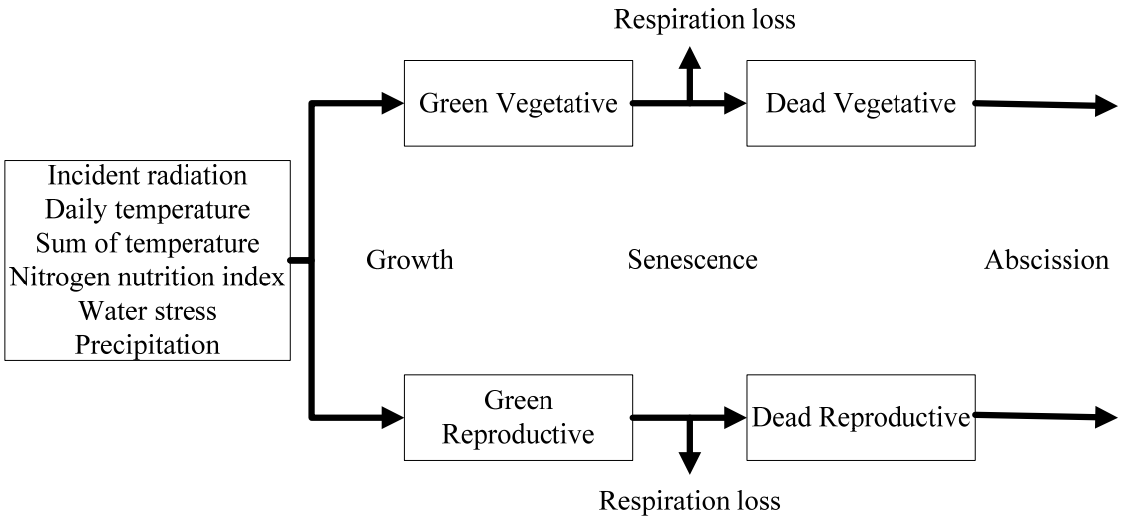


Chart 2-2 Biomass flow within the life cycle of the grassland [1]

Chart 2-3 shows the relationship and the diversion of the variables in the compartment GV and GR.

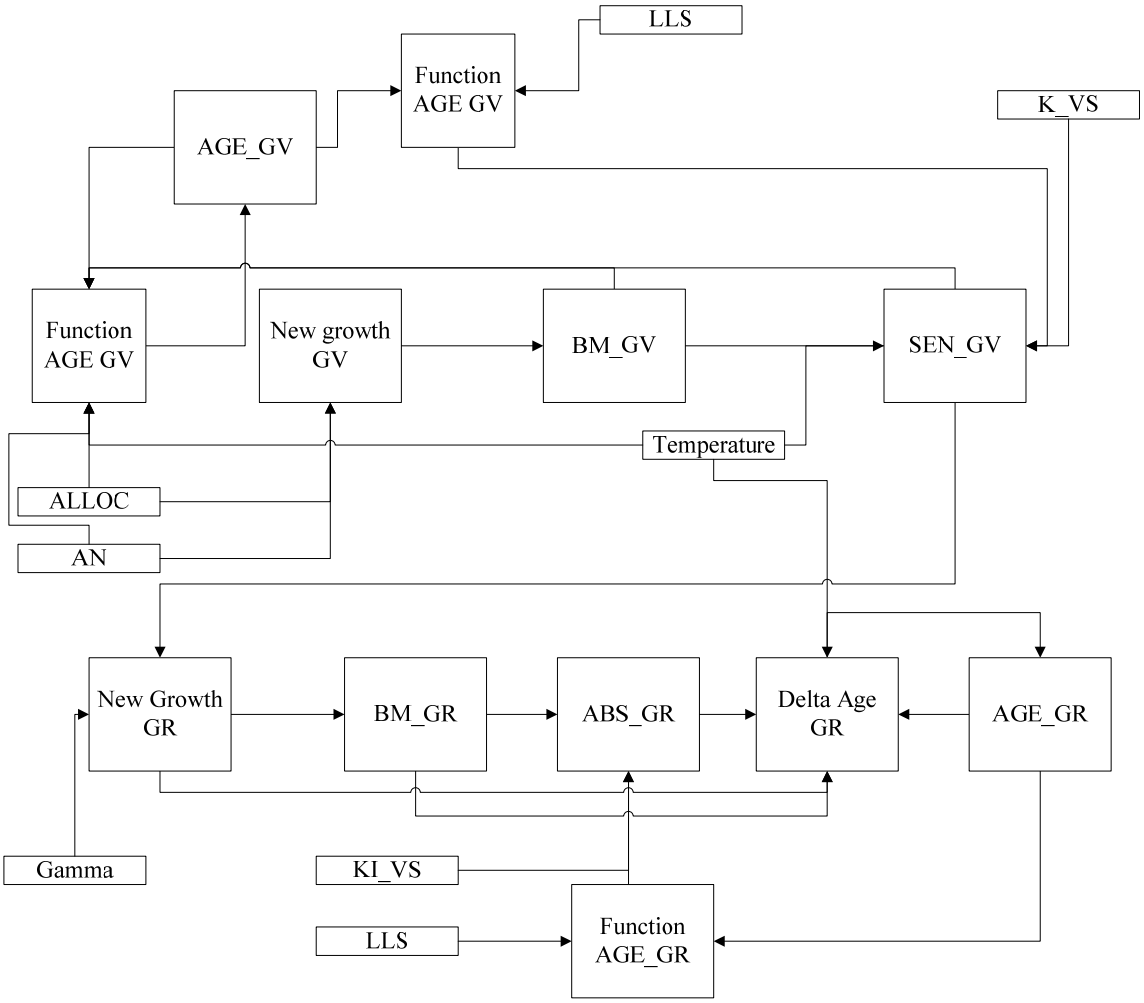


Chart 2-3 Variables repartition flow of compartment GV and GR

From the flow above, there are some processes to compute the biomass. BM_GV and BM_GR are the variables as the final output but they are also used in the processes to compute other variables. In fact, the initial biomass can be evaluated by the practical experiments or from the previous computation. For example, the biomasses of day 1 are already computed as BM_GV = A and BM_GR = B. These two variables will take part into the process of computing the biomasses of day 2. Like this kind of process, the values of biomasses are reused and updated time by time. AGE_GV and AGE_GR are in the same situation. Delta Age GV and Delta Age GR are the processes which reuse the previous values of AGE_GV and AGE_GR and update the values of AGE_GV and AGE_GR.

Some other variables are not input data neither output data. They correspond to the intermediate results, such as ALLOC, AN, etc. The biological basis for the growth functions is described in the papers of Jouven et al (2006) [1]. Step by step,

the processes of computation show below. These constraints and equations are presented and used in Berkeley Madonna to simulate the vegetation growth model.

1) To compute AN, the equation is as follows.

(1) AN = ENV*PGRO*SEA

Then we need to compute ENV, PGRO and SEA.

(2) PGRO = PAR*RUEmax*(1-exp(-k*LAI))*10);

(3) F Temperature =
MAX(0,MIN(1,(Temperature-TB1)/(10-TB1)));

(4) F PAR = MIN(1,1-0.0445 * (PAR-5));

(5) W = WR/WHC;

To compute W, Chart 2-4 shows the variables needed and the diversion of these variables.

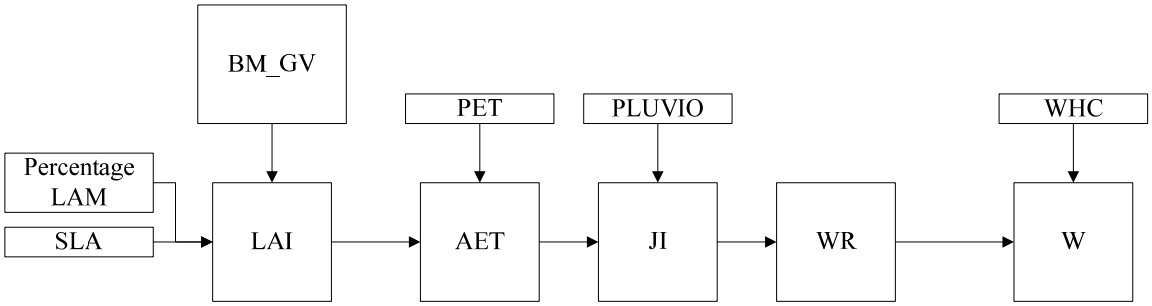


Chart 2-4 Flow computing W

(6) LAI = SLA*Percent_LAM*B_VG/10

(7) AET = MIN(PET, PET*LAI/3)

(8) J1 = PLUVIO-AET

W is the variable showing the water stress to compute the ENV. When W is computed by the flow above, F_W should be computed by contrasting the value of PET. For easily understanding, the process of contrasting is showed in the way of pseudo code.

(9)

```
IF(PET <= 3.8)
    IF(W<=0.2)
        F_W = 4 * W
    ELSE IF (W <= 0.4)
        F_W = 0.75 * W + 0.65
    ELSE IF (W <= 0.6 )
        F_W = 0.25 * W + 0.85
```

```
    ELSE
        F_W = 1;
ELSE IF(PET <= 6.5)
    IF(W<=0.2)
        F_W = 2 * W
    ELSE IF (W <= 0.4)
        F_W = 1.5 * W + 0.1
    ELSE IF (W <= 0.6 )
        F_W = W + 0.3
    ELSE IF (W <= 0.8 )
        F_W = 0.5 * W + 0.6
    ELSE
        F_W = 1;
ELSE
        F_W = W
END.
```

Then continue to computing the other variables.

(10) F IN = 0.25+(0.75*(IN-0.35)) / 0.65

(11) ENV = F_PAR*F_W*F_IN*F_Temperature

To compute SEA, the pseudo codes are as following.

(12)

```
IF(SUM_Temperature < 200)
    SEA = Min_SEA
ELSE IF ( SUM_Temperature < (ST1-200))
    SEA = Min_SEA+(Max_SEA-Min_SEA)*
        (SUM_Temperature-ST1+200)/100
ELSE IF (SUM_Temperature<ST1-100)
    SEA = Max_SEA
ELSE IF (SUM_Temperature<ST2)
    SEA = Max_SEA+(Min_SEA-Max_SEA)*
        (SUM_Temperature-ST1)/(ST2-ST1)
ELSE
    SEA = Min_SEA
END.
```

2）To compute ALLOC, the equation is as follows.

(1) ALLOC = F_IN*Cut

To compute Cut, the pseudo codes are as following.

(2)

```
IF(SUM_Temperature < ST1)
    Cut = 0
ELSE IF (SUM_Temperature > ST2)
    Cut = 0
ELSE
    Cut = 1
```

Then BM_GV and BM_GR are computed as following.

3）To compute BM_GV, the equations are as below.

(1) BM_GV + = new growth GV

(2) new growth GV = An*(1-ALLOC)

4）To compute BM_GR, the equations are as below.

(1) BM_GR + = new growth GR

(2) new growth GR = (1-gamma)*sen_gv

To compute sen_gv, the pseudo codes are as following.

(3)

```
IF(Temperature>0)
    sen_gv = K_VS*BM_GV*function_age_GV*Temperature
ELSE
    sen_gv = 0
```

The pseudo codes below are to compute function_age_GV.

(4)

```
IF (AGE_ GV /LLS < 1/3)
    function_age_GV = 1
ELSE IF (AGE_ GV /LLS < 2/3)
    function_age_GV = 3*AGE_ GV /LLS
ELSE
    function_age_GV = 3
```

AGE_GV is computed by following equations.

(5) AGE_GV + = Delta age GV

(6) Delta age GV = ((BM_GV-sen_gv)
/(BM_GV-sen_vs+(An*(1-ALLOC)))*(Age_GV+Temperature)-Age_G
V)

Additional variables are needed to compute the AGE and abscission of compartment GR.

To compute function_age_GR:

(7)

```
IF (Age_GR/LLS < 1/3)
    function_age_GR = 1
ELSE IF (Age_VS/LLS < 2/3)
    function_age_GR = 2
ELSE
    function_age_GR = 3
```

To compute abscission GR:

(8)

```
IF(Temperature>0)
    ABS_GR = Temperature*Kl_VS*B_VS*function_age_GR
ELSE
    ABS_GR = 0
```

To compute AGE_GR:

(9) AGE_GR += Delta_age_gr

(10) Delta_age_gr = (BM_GR-abscission_GR)
/(BM_GR-abscission_GR+new_growth_gr)*(Age_GR+Temperatur
e)-Age_GR

Compartment DV and DR have a similar structure to compartment GV and GR but there are some differences in using different variables, equations and conditions. Chart 2-5 shows the relationship of the variables in the compartment DV and DR. Because of the similarity in the structure and equations, the formulas which explain the method to compute the variables in the compartment DV and DR are not declared in detail.

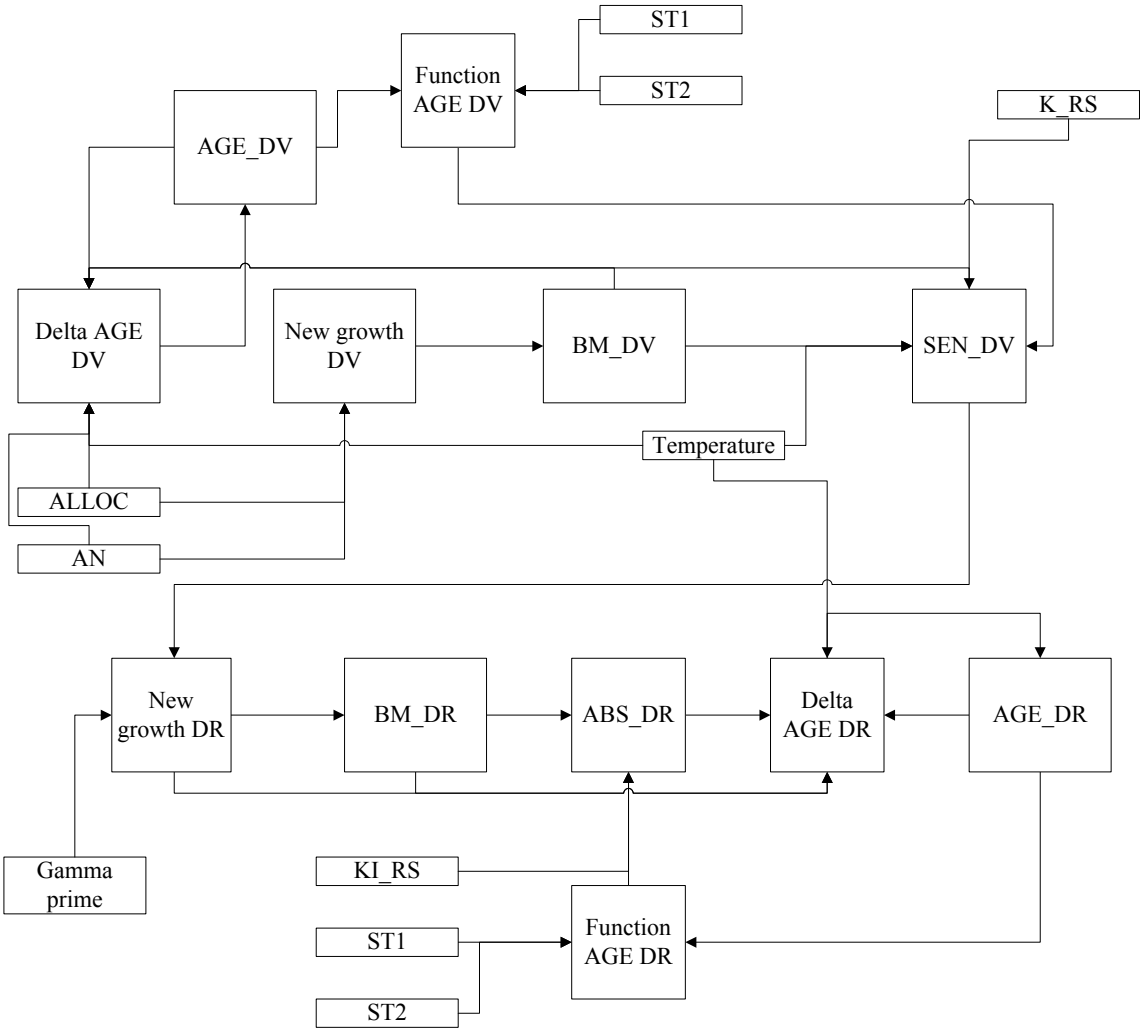


Chart 2-5 Variables repartition flow of compartment DV and DR

2.2 Brief summary

This chapter presents methods which transfer the biologic rules to the language of computer science and describes main functions of this model. First, this chapter presents the logic structure of the pasture then specifically illustrates methods of computing the biomass of these four compartments with the flow charts and pseudo codes.

Chapter 3 System Requirement Analysis

3.1 General system requirements

The task of this chapter is to analysis the requirements for the model described in Chapter 2. This chapter focuses on the computer science method which presents demands of the system supporting the model. This system implements functions which are described by the model and receives the input information from users. After computation following the rules of the model, the system will record the final result into the files and display it on the interface. Chart 3-1 shows the use-case diagram of the system.

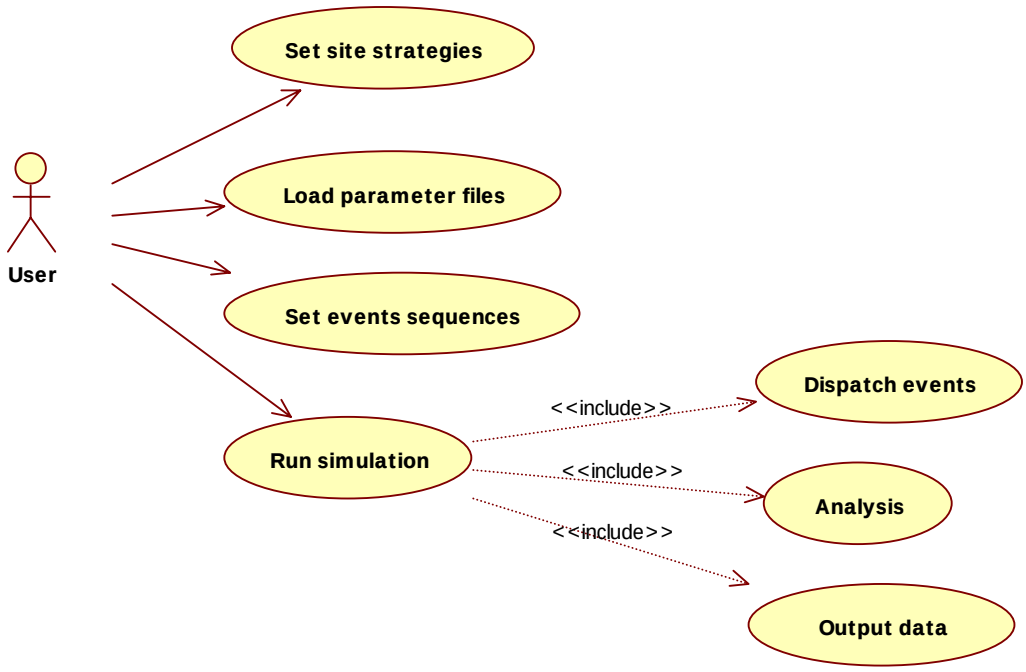


Chart 3-1 Use-case diagram of the system

For initializing the grassland, several patch (vegetation) types are used to determine the initial traits of cells. Users firstly decide site distribution and the patch type of these sites. Then users load files which contain all the parameters needed to initialize traits of patches and environment variables. Next, they determine sequences of events (event types and functions are described in “3.2.2 Events description”). When everything is ready, users run simulations and get

results. During the time that the software is running, actions are done in three steps. First, the system dispatches events which are determined by users. Second, when certain types of event are executed, the data corresponding to this type event is analyzed by the system and updates the data pool. Finally, when data is successfully updated, system outputs data into output files and shows data on the interface.

3.2 The functional requirements

3.2.1 Vegetation phase

Main methods of this phase are already described in Chapter 2 and the task of software is to implement these methods to program and use them to compute the biomass of the pasture. So the requirements of the vegetation phase is to implement the model which is described in Chapter 2 and handle the input and the output of the system.

All the input variables can be measured by field measurements or practical experiment. For every compartment, the amount of biomass compartment can be computed as a function of time. Table 3-1 shows the input and the output of the system in biological terms. This table illustrates all the possible inputs, the intermediate results and all the outputs needed but not the exact relationship between the inputs and the outputs.

Table 3-1 System input, intermediate result and output in the vegetation phase

Input	Intermediate result	Output
Gamma	AET	DV_BM
Gamma prime	AGE	DR_BM
IN	ALLOC	GV_BM
K_VS	AN	GR_BM
K_RS	CUT	Sum BM
KI_VS	ENV	Mean_BM
KI_RS	LAI	Standard deviation_BM
LLS	PGRO	
PAR	SEA	
PET		
Percent LAM		
PLUVIO		
RUEmax		
ST		

3.2.2 Event description

To simulate real situations of the grassland, some actions of agricultural management should be considered. In this system, these actions of agriculture are determined as events. So the process of simulation is creating events, dispatching events and executing events. Events involved into this system are as following.

1. Cut: “Cut” means cut all the plants on the grassland and considered as a harvest action. There are some constraints of cut in the system.
   (1) Cut date is defined at start in the interface.
   (2) Biomass is considered to be cut X m above ground level in all cells on that date. The default value is 0.05 m in line with local management practices.
   (3) Residual biomass in each structural compartment is calculated using vegetation bulk densities (BD). For example, for compartment GV:
   Res biomass VV = X * 10 * BD VV.
   (4) BD values are fixed for each patch type (need to be included in initial parameter file).
   (5) Harvested biomass = Standing biomass – Res biomass

(6) Total harvested biomass = Sum of harvested biomass for each compartment (GV, GR, DV, DR).

(7) When cut happens, need to modify CUT from 1 to 0 to affect reproductive growth (REP function), modify biomass in each compartment and store values for harvested biomass.

2. Fertilization: As suggested by its name, fertilization means the application of inorganic nitrogen-based fertilizer to the grassland. The effect of fertilization is not immediate influence on the grassland but a constant effect. The rules of fertilization are as following.

   (1) Fertilization date is defined at start.

   (2) Inorganic fertilizer is considered to be applied to all cells on one date.

   (3) Fertilizer is assumed to increase NI after 7 days where water is non-limiting i.e. W > threshold value (to be defined in parameter file).

   (4) Fertilizer is assumed to be sufficient to shift NI status of cell from low (0.55) to medium (0.75), from medium to high (0.85) and from high to non-limiting (1).

   (5) In the absence of fertilizer addition, NI is assumed to stay constant over course of 1 year

3. Grazing: Grazing refers to the presence of domestic herbivores on the grassland, which graze the grass and put excreta (named as "Dung" in the system) on the grassland. The grazing event is composed of 2 parts, defoliation and dung.

   (1) Defoliation: Animals will graze swards and movements of animals will cause defoliant of swards. The rules of defoliation are described as following.

      A. Start date is defined for grazing (GRAZ = 1) and the same as finishing date.

      B. Biomass is considered to be removed daily for a proportion of cells (proportion to be defined in a parameter file, dependent on animal stocking rate)

      C. For each grazed cell, biomass is considered to be removed to 0.05m above ground level. . Residual biomass is calculated as above and CUT function is modified (ingested biomass can be calculated and stored as 'Cut').

D. Cell selection for grazing is not random and there are some rules to set priorities of defoliation actions.
    a. Defoliation only occurs for cells with > minimum biomass in GV and with > minimum GV/GR ratio (values to be defined in parameter file).
    b. Defoliation will not occur for cells with > maximum biomass in DV and DR (values to be defined in parameter file).
    c. Defoliation occurs preferentially for particular patches.
    d. Defoliation occurs preferentially for cells with a high NI.
    e. Defoliation will not occur for cells with dung or for cells immediately next to dung (size of buffer zone to be defined in parameter file, cells need to be flagged)

(2) Dung: Dung refers to solid animal returns discharged on the grassland during the presence of animals. These wastes fertilize swards as fertilization, but to a lesser extent because they provide an organic rather than inorganic source of nitrogen. There are some disciplines to constrain actions of Dung as following.
    A. Dung is assumed to occur daily for GRAZ = 1.
    B. Number of dung patches depends on stocking rate (10 per day per animal): define animal number at start.
    C. Size of dung and dung buffer zone is defined at start (default value = 3 * 3 cells for dung with 3 cells buffer strip).
    D. Spatial distribution of dung patches is negative binomial.
    E. Assume that dung patch stops growth of cells under dung during x weeks (default value = 12 weeks), reduces cell biomass to a minimum value (to define) and stops reproductive growth (use CUT function).
    F. Assume shift NI status of cells influenced by dung after 7 days where water is non-limiting i.e. W < threshold value (to be defined in parameter file).
    G. Dung is assumed to shift NI status to 1/3 that of fertilizer increases (i.e. +7 rather than +21) for cells under dung and in buffer zone.

3.3 Additional requirements

1) Handling the numerous data of the pasture effectively.

2) Collecting the information needed to predict the biomass production and the structure of the pasture.

3) Illustrating the result of predictions and accumulations of biomass of the pasture over the year/growing season.

4) Simplifying the input data of the model.

5) Output data format can be produced as NetCDF (Network Common Data Form) which is a kind of file format for science computation and analysis and CSV (Comma Separated Values) which is another file format to show results in a simple way.

3.4 Brief summary

This chapter describes the general system requirements, functional requirements and additional requirements. It provides its basic structure and describes its features. It ensures all the functions needed to implement of this system and the demands of the graphic user interface which maintains and runs the agriculture analysis model.

Chapter 4 System Design

The aim of this chapter is to present the design of the system according to the requirement described by chapter 3. System logical structure design is given first to illustrate whole system. Then the structure of logic module, GUI module and IO part are described.

4.1 System logical structure design

This system consists of 2 functional modules: GUI (graphic user interface) module and Logic module. IO provides methods to load file or output final data to files. IO part cannot exist without functional modules and methods of IO part are called and used by those two functional modules. The system structure is illustrated by Chart 4-1.

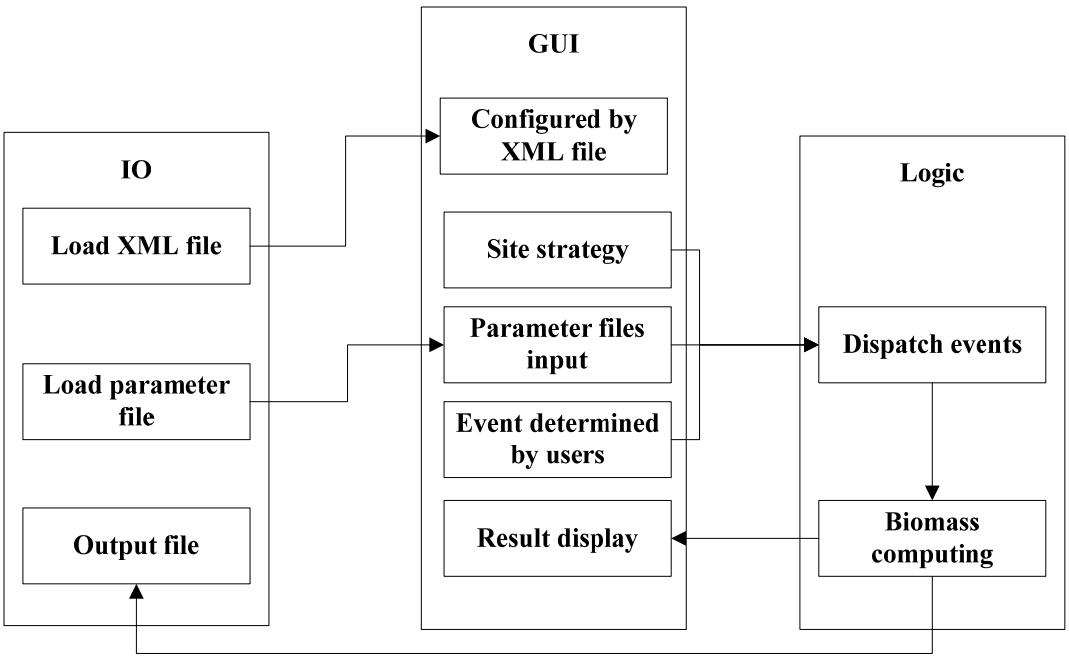


Chart 4-1 System logical structure

GUI module implements an interface which accepts manipulations of users and shows results of simulations. The variables of GUI can be configured by XML file. XML file is used to load user input data, site setting data, paths of input parameter files and format of output files (CSV, NetCDF or the both). Site strategy setting helps users to determine the structure of pasture and the distribution of patches.

Parameter file paths can be inputted into the XML configuration data file or given by users' manipulation. Event sequences are the same as parameter files. They can be fulfilled by the XML configuration file or via GUI. When a simulation is finished, output data is showed in the final result table and there is a special screen which displays the situation of each cell. Logic module executes events according to the event sequence. When executing each event, the computation of biomass is done according to the type of events.

4.2 IO part specific design

IO part is a method pool which provides input and output methods for GUI module and Logic module. Input tasks include inputting information files which files' type is CSV and inputting XML configuration files. Output tasks include outputting XML configuration files, outputting result data in CSV file format and outputting result data in NetCDF file format.

4.2.1 Input specific design

There are 5 kinds of information files which provide different and necessary data to the system. All these files are in CSV format and are used as the initial input of the system to initialize variables of the pasture.

1) Constant file: Contain constant variables of soil and radiation. The specific description of each variable is illustrated in table 4-1.

Table 4-1 Variables of Constant file

Variables	Description	Unit
S	Cell area	m^{-1}
RUEmax	Maximum radiation use efficiency	g MJ^{-1}
Sigma	Respiratory C loss during senescence (GR)	
Sigma prime	Respiratory C loss during senescence (DR)	
Hrv lim	Height of RV	m
Int defol seuil	Defoliation threshold	

2) Environment file: Contain environment variables which are practical measured data in one year at a daily time step. The specific description of each variable is illustrated in table 4-2.

Table 4-2 Variables of Environment file

Variables	Description	Unit
Jour	Day	day
Temperature	Average temperature of each day	°C
PARi	Incident photosynthetically active radiation	MJ m^{-2}
PP	Precipitations	
PET	Potential evapotranspiration	mm

3) Parameter file: Contain parameters which are the information of scales of pasture and overall patch type. The specific description of each variable is illustrated in table 4-3.

Table 4-3 Variables of Parameter file

Variables	Description	Unit
Nombre de lignes	Number of lines	
Nombre de colonnes	Number of columns	
Nombre de facies	Number of patches	

4) Cel-site file: This structure of this file depends on the dimension of the plot and the distribution of patches. Each “table cell” of the table in this file records the patch type which corresponds to the plot cell of the same index in plot (i.e. if the cell line x column y of the file contains the number z, then it means that the cell (x, y) in the plot belongs to patch z). This file can be considered as a digital representation of the plot.
5) Patch file: Contain all the variables which describe traits of patches. The specific description of each variable is illustrated in table 4-4.

Table 4-4 Variables of patch file

Variables	Description	Unit
ST1	Onset of reproductive growth	Degree/day
ST2	End of reproductive growth	Degree/day
INcell	Nutritional index of cell	-NNI
Wcell	Cell biomass	kg ha-1
Hcell	Height of cell	m
WHC	Soil water-holding capacity	mm
minSEA	Growth increase in winter	
maxSEA	Growth increase in summer	
W_VV	Biomass of GV	kg ha-1
alphaPAR	Light extinction coefficient	
T0	Temperature threshold: photosynthesis activation	°C
T1	Temp threshold: stable growth	°C
T2	Temp threshold: growth decline	°C
B_IN	Impact of IN on LUE at IN=0	
SLA	Specific leaf area	m2 g-1
LLS	Leaf lifespan	Degree/day
Rho_VV	Volume GV	g m-3
Gam_min	% leaf lamina (min theshold)	
Gam_max	% leaf lamina (max theshold)	
gammaVV	% leaf lamina GV	
W_RV	Biomass of DV	kg ha-1
A_IN	Value of ALLOC at IN=0	
Rho_RV	Volume RV	g m-3
W_VS	Biomass of VS	kg ha-1
K_VS	Senescence coefficient VS	Degree/day
Kl_VS	Abscission coefficient VS	Degree/day
Rho_VS	Volume VS	g m-3
W_RS	Biomass of RS	kg ha-1
K_RS	Senescence coefficient RS	Degree/day
Kl_RS	Abscission coefficient RS	Degree/day

Beside these information files, XML configuration file is used as recording the configuration data of one simulation. The structure of XML configuration file is as following.

```
<configuration>
    <GeneralParameters>
        <StartDay>Number of start day</StartDay>
        <Duration>Number of duration</Duration>
    </GeneralParameters>
    <InputFiles>
        <ParameterFile>Path of parameter file</ParameterFile>
        <PatchFile>Path of patch file</PatchFile>
        <EnvironmentFile>Path of environment file</EnvironmentFile>
        <ConstantFile>Path of constant file</ConstantFile>
        <CelSiteFile>Path of cel-site file</CelSiteFile>
    </InputFiles>
    <Event>
        <EventName>Event name</EventName>
    </Event>
    <Output>
        <OutputCSV>True or false</OutputCSV>
        <OutputNetCDF>True or false</OutputNetCDF>
    </Output>
</configuration>
```

4.2.2 Output specific design

According to the demands, two types of outputting files are needed to store simulation result. The first kind of file contains the final biomass for each cell. The second kind one records the temporary biomass of the entire pasture. In the last, the day number, sum of biomass in the plot, the minimum and maximum biomass for a cell in a plot and standard deviation of biomass between all the cells of the plot are recorded at a daily time step to show the vegetation dynamic during the year.

4.3 GUI module specific design

Graphical User Interface module is the only module which is in charge of communicating with users and showing them results. Following user demands, the aim of the GUI design is focused on providing a convenient control of inputting and showing the result in a graphic way. So to be user friendly and as simple as possible, the interface is designed as a common Windows program. Each function is arranged in different tab, ordered like this: To clearly classify functions, the method of interface arrangement is as following.

1) Menus: it contains main functions concerning the manipulation of XML files and selecting output file types.
   - Menu File: Save actual configuration to a XML file already created or not; load a XML file in order to parameterize a simulation
   - Menu Run: Run loaded simulation or select output file format (CSV, NetCDF or both).
2) General Tab: Allow to specify the start day and the duration of simulation.
3) Set patch Tab: There are four methods to set patches to cells.
   - Manual: Users decide on the dimension of the plot by inputting the number of cells in vertical and horizontal directions. An initial simulated plot is drawn on the interface with the dimension the users enter. Then users choose a patch type and use a mouse to draw a rectangle. The entire cell in the rectangle switches their patch to the chosen patch. Users repeat the operation until they decide to finish deploying patches, click on save button to store the result into “Cel-site” files.
   - Random: Firstly, users decide the percentage of each patch type. The sum of these percentages has to be naturally 100. Then users click on the launch button to get the result from the interface. The patch type randomly deploys onto cells according to the percentage.
   - Aggregate: A Poisson distribution is used in this method. Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event. The Poisson distribution can also be used for the number of events in other specified intervals

such as distance, area or volume. If the expected number of occurrences in this interval is λ, then the probability that there are exactly k occurrences (k being a non-negative integer, k = 0, 1, 2, ...) is equal to

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

where

A. e is the base of the natural logarithm (e = 2.71828...)
B. k is the number of occurrences of an event — the probability of which is given by the function
C. k! is the factorial of k
D. λ is a positive real number, equal to the expected number of occurrences during the given interval. For instance, if the events occur on average 4 times per minute, and one is interested in the probability of an event occurring k times in a 10 minute interval, one would use a Poisson distribution as the model with λ = 10×4 = 40.

The first step is the same as the random method but the aggregate method used a radius around the points which is generated by Poisson distribution as the input variable. The process of aggregate method is as following.

A. Randomly generate one point following Poison distributed constraints.
B. Set the cell patch according to the percentage of patches given before.
C. Once the center point is chosen in the first step, a rectangle is created around it which a fixed radius and the same patch is deployed in all the cells which belong to this rectangele.

- Load from file: Load the patches setting strategy from the file “Cel-site” and expose it on the interface.

4) Graphic output Tab: This tab shows details of biomass distribution of each compartment. Cells are showed as polygon and are colored depending on their biomass relative to the average biomass standard.

5) Final result Tab: This tab contains a table which presents the summary of final result, including: sum, minimum, maximum and standard deviation of biomass for total compartment and each compartment respectively.

Chart 4-2 illustrates the UML class diagram of GUI module. It describes all the classes, main attributes and major methods of GUI module.

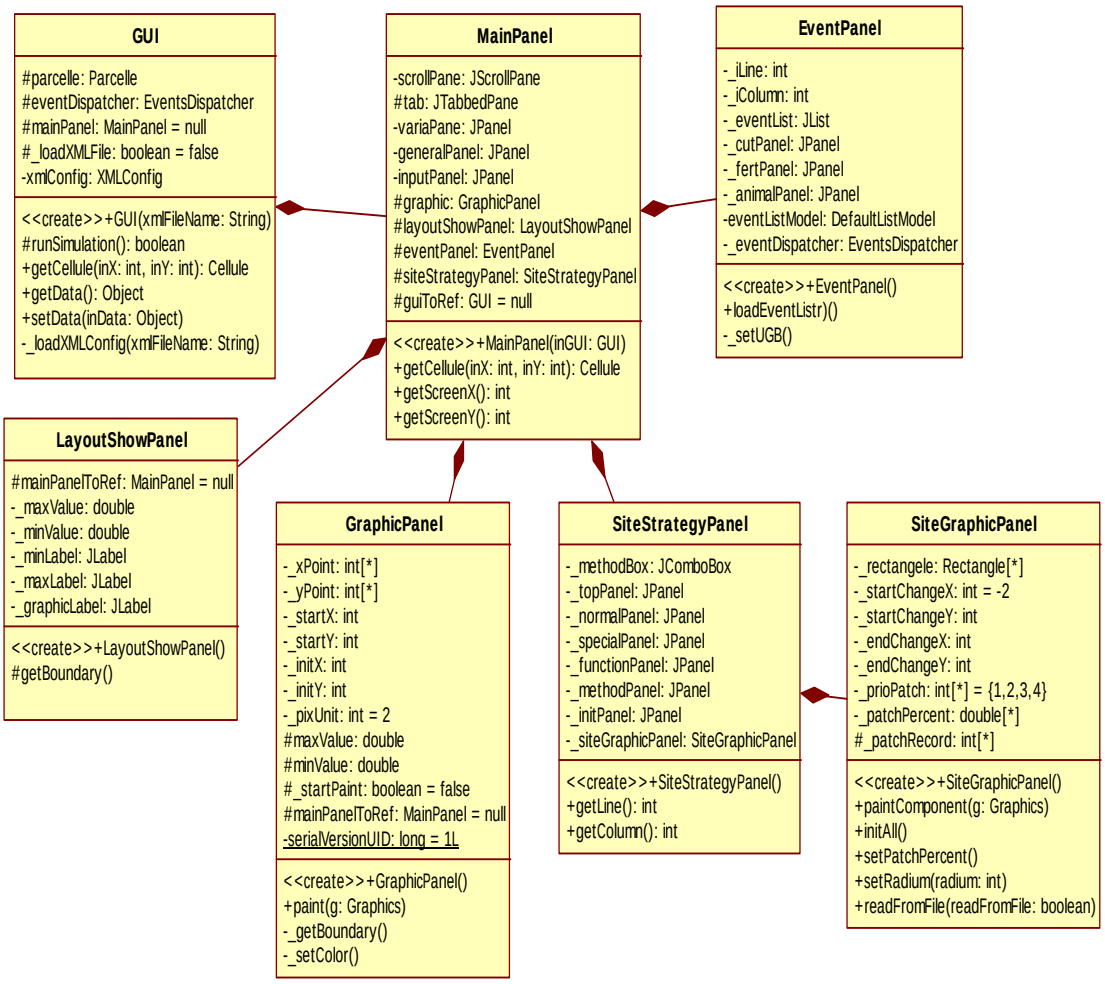


Chart 4-2 class diagram of GUI module

1) GUI class: Top class of GUI module. It instigates the MainPanel object and Logic module objects. It generates the main frame of the GUI and allows loading XML files which are used for initializing the input simulation.
2) MainPanel class: This class creates the main panel which is contained by the main frame. It encompasses the other classes of GUI module except GUI class. The structure of tabs is initialized by MainPanel. The location of

the interface in the screen is in this class as well. It submits the location of the interface to the graphic painting class in order to correct the start painting point for each graphic panel.

3) EventPanel class: This class generates the interface of event management. It consists of a list which represents the event sequences. Users are able to create a new event by choosing the event type and fill the information concerning this event. Clicking on some event on the event list, users are able to modify that event or delete it. For modification, the detailed information of that event is shown in the interface. The variables of each kind of event are described as following.
   (1) Cut event: Cut start time and height of grasses after cutting.
   (2) Fertilization event: Fertilization start time
   (3) Grazing event: Graze start time, graze end time and stocking rate. The stocking rate describe the number of animals that are present in the plot. Stocking rate means that how many animals are distributed in some scale of grassland. Its unit is lu/ha which lu (Livestock Unit) stands for a constant rate depending on the type of animals. For example, a dry medium cow means 0.8 lu. If there are 10 of these cows in a grassland of 10 hectares, the stocking rate is 0.8*10/10 = 0.8 lu/ha. In the system, 1 hectare means that there are 100,000 cells.
4) LayoutShowPanel class: This panel layout is a bar under the graphic panel which compute basic statistics like minimum and maximum value of the biomass for one compartment.
5) GraphicPanel class: Final results of biomass for each cell are shown in this panel in a graphical way. Each cell is painted as a polygon and colored depending on its final biomass among the average biomass standard.
6) SiteStrategyPanel class: Users can set the dimension of the plot and coupling each cell to a patch type via an interface. Users just choose which method is used and enter corresponding information in order to create the patch-setting strategy. This strategy will be shown in the interface.
7) SiteGraphicPanel class: This class is contained by SiteStrategyPanel class. It implements a panel which displays patch deployment strategy according to the users input on the site strategy panel or from XML file.

4.4 Logic module specific design

The logic module deals with event dispatching and biomass computing by using vegetation growth model. Input of this module comes from GUI and IO method. Output is displayed via the GUI and stored files (CSV and/or NetCDF format). Relationships between the GUI module and the logic module are illustrated by chart 4-3.

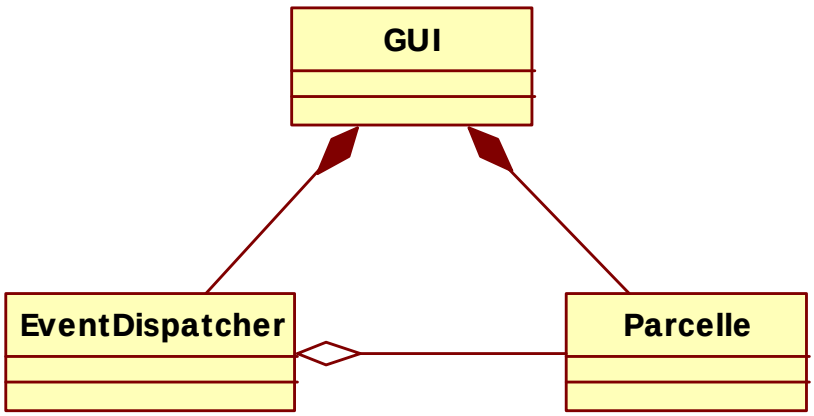


Chart 4-3 UML class diagram which illustrates relations between GUI and Logic

EventDispatcher and Parcelle are main classes of Logic module. EventDispatcher is in charge of managing event sequences and Parcelle receives data from input files and computes the biomass according to the present event type. They are all encompassed by GUI class, receiving input data from GUI and transferring results to GUI for displaying or storing output data to files.

4.4.1 Event dispatcher specific design

Based on user demands, there are 5 kinds of events which should be considered in this part.

1) Vegetation event: This event has low priority so it is executed after all other events occur. This event updates the vegetation (biomass, age of compartments).
2) Cut event: Only occur at the time of “Cut start day”. At this time, grasses in the plot are cut at the height given in the parameter.
3) Fertilization: This event only occurs at the time of “Fertilization start day”. The effect of fertilization influences NI value for a long period.

4) Grazing: This event starts at the “Grazing start day” and finishes at the “Grazing end day”. The distribution of grazing is according to the priority of cells. According to the demands of grazing, the rules of priority are given as following.

(1) Eliminate ungraspable cells.

In order to allow defoliation, cells must have a vegetation height taller than 0.05 centimeter (the height is calculated using BD as with “Cut” event) and biomass of GV compartment must be greater or equals to 50% of the total cell biomass. Cells must not have dung or be flagged as the dung buffer zone. Here are the cases where there should be no defoliation:

A. Cell with dung (or flagged as dung buffer zone)

B. Cell with (DR + GR) biomass greater than fifty percent of total cell biomass

C. Cell where vegetation height equals or less to 0.05 centimeter.

(2) Set the priority rules.

The priority is classified into 4 levels. To calculate the priority 1-4 for cells which meet the defoliation criteria, we take into account: patch type, nutritional index and proportion of vegetative green biomass (GV). Each factor has a score. Table 4-5 shows the rules of setting scores.

Table 4-5 Rules of setting priority scores

Patch type	Score	IN	Score	%GV	Score
A	4	No limitation/High	4	70	4
B	3	Medium	2	60-70	2
C	2	Low	1	50-60	1
D	1				

The total score for the cell is calculated by multiplying the score of the cell patch, IN and %GV. Example: A cell is patch type A, with medium IN and %GV of 65. The total score is 4 * 2 * 2 = 16

Total score determines the grazing priority.

Priority 1: Total score >30

Priority 2: Total score >15

Priority 3: Total score >7

Priority 4: Total score <7

5) Record temporary data to files: Because of the memory limitation, it is impossible to record all the data in the memory. Temporary data should be recorded in files and that is only the latest data which will be recorded in the memory. This kind of event is happened everyday but always executed in the end.

The structure of event dispatcher part is showed in chart 4-4 UML class diagram of event.

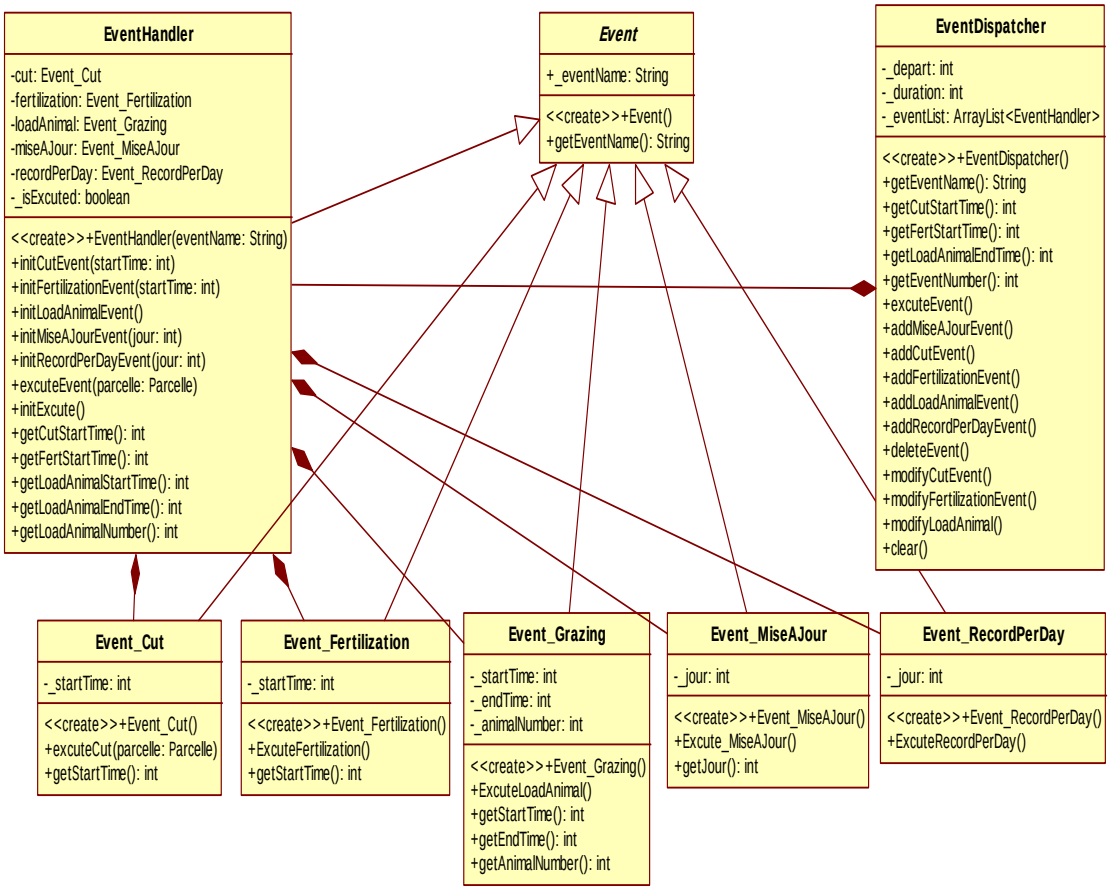


Chart 4-4 UML class diagram of event

- Event class: Event class is an abstract class which is the super class of Event_Cut class, Event_Fertilization class, Event_Grazing class, Event_MiseAJour class, Event_RecordPerDay class and EventHandler class. It contains an attribute and a method. The attribute “eventName”

is the public attribute for all the children classes and it is the only attribute to classify children classes.

- Event_Cut class: This class extends Event class and implement the method of "Cut" event. Method excuteCut aggregates a Parcelle object to analyze the biomass through the method in Parcelle.
- Event_Fertilization class: Extends Event class and implement the method of "Fertilization" event. Method excuteFertilization aggregates a Parcelle object to analyze the biomass through the method in Parcelle.
- Event_Grazing class: Extends Event class and implement the method of "Grazing" event. Method excuteLoadAnimal aggregates a Parcelle object to analyze the biomass through the method in Parcelle.
- Event_MiseAJour class: Extends from Event class and implement the method of "Vegetation" event. Method excuteMiseAJour aggregates a Parcelle object to analyze the biomass through the method in Parcelle. This event is executed every day during the period of simulation.
- Event_RecordPerDay class: Extends from Event class and implement the method of the event which records the temporary data for per day. Method excuteRecordPerDay aggregates a Parcelle object to analyze the biomass through the method in Parcell. This event is executed every day during the period of simulation.
- EventHandler class: Extends from events and able to generate an object for Event_Cut, Event_Fertilization, Event_Grazing, Event_MiseAJour and Event_RecordPerDay. For managing events, event dispatcher utilizes an array list to contain different event objects but because of the technical reason, there should be only one kind of object contained in the array list. The purpose of the EventHandler class is to ensure the uniqueness of classes in the arraylist. At the same time, it initializes the other event objects and provides methods necessary for the other events. It classifies different event objects by knowing event names.
- EventDispatcher class: This class composes EventHandler class. There is an array list which records event sequences implemented in this class. This array list binds EventHandler object to generate the other type of objects. EventDispatcher class gets names of events and implements

adding method, deleting method and modifying method for all the events.

This structure of class diagram can be considered as a kind of event-driven programming. The flow of system is determined by events and all the actions and analysis of the system is based on events type. All the actions are encapsulated into events and system just deals with their execution. It simplifies the structure of the system and enhances its stability. Superior hierarchy applications do not need to be in charge of the details of methods of inferior hierarchy program but just dispatch event sequences and execute events.

4.4.2 Vegetation specific design

The vegetation part is in charge of analyzing input data, generating intermediate results and outputting the final results. This part is designed with a hierarchy structure. The foundation part is the pasture which stands for all the information of the grassland including: environment variables, constant variables, patches variables and patch distribution variables. The pasture consists of many cells. Each cell has 4 compartments and a patch type which is used as initializing the variables of the cell. There are 4 types of patches in the system and variables of each patch cannot be changed during the simulation. Chart 4-5 illustrates the class diagram of the vegetation part.

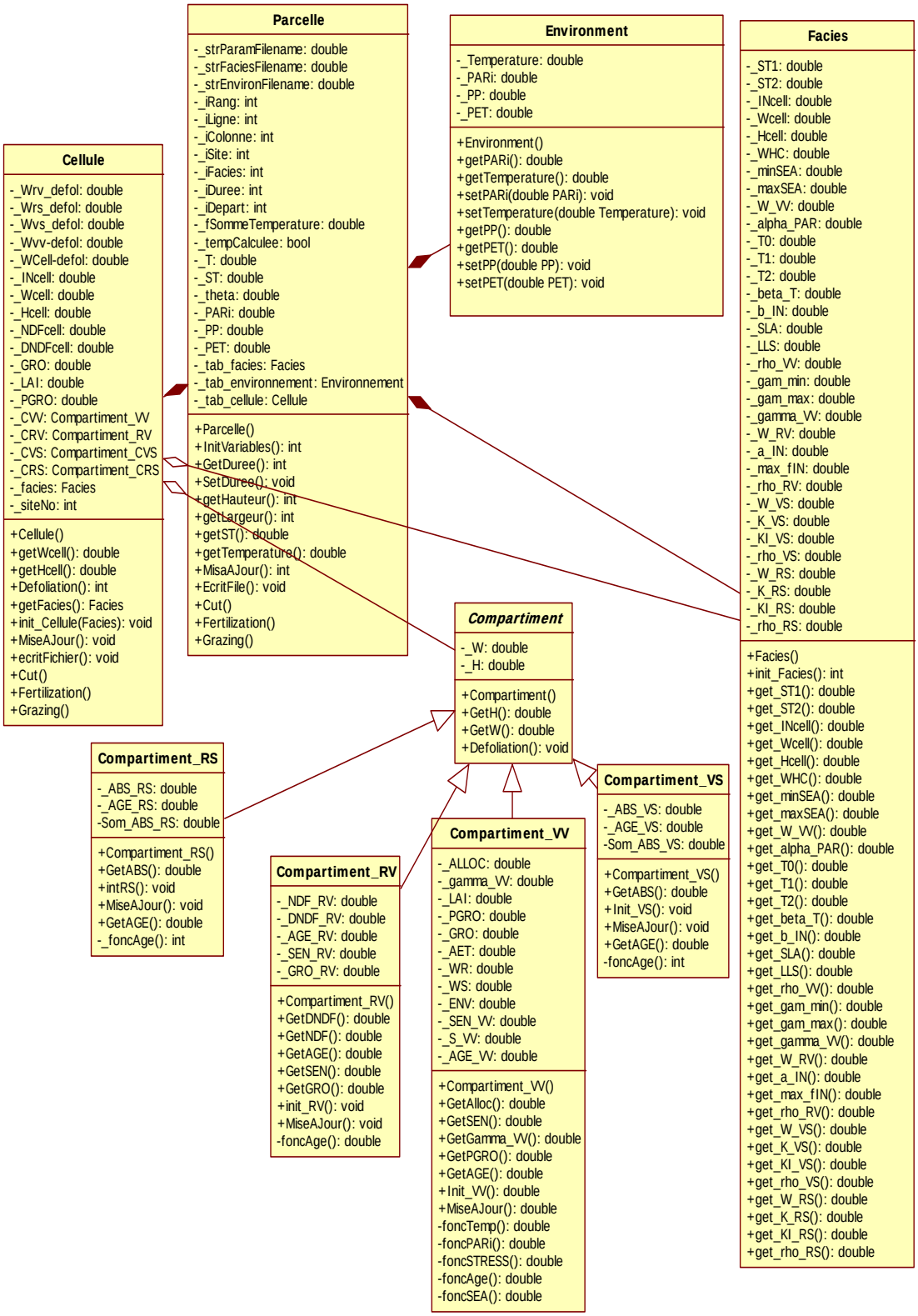


Chart 4-5 Vegetation class diagram

Because this project comes from a French laboratory, the class names use French words. The main functions of each class are described as following.

1) Parcelle class: Parcelle, which stands for pasture, is the superior class of vegetation part. It communicates with event part and GUI modules. Event part and GUI modules just aware methods of Parcelle class but not the other classes of the vegetation part. It keeps the encapsulation of the structure of vegetation part. It composes cellule class, environment class and facies class. MiseAJour method means updating the situation of pasture and computing vegetation growth of the pasture corresponding to the vegetation growth model for each day during the period of simulation. Cut method is used as computing the biomass after cut. Fertilization method is used to change the variables of NI after fertilization. Grazing method is executed during the period of grazing and changes the distribution of grazing based on the distribution of dung. Initvariables method initializes all the variables needed in the Parcelle class. EcritFile method outputs the result data to the final result files by CSV format or NetCDF format.
2) Environment class: This class includes all the variables of environment data. It just provides the getting and setting methods to obtain and modify the value of these data. These methods are called in the Parcelle class. It is used to enhance the encapsulation of the system.
3) Facies class: Facies is a French word standing for vegetation patch type. So this class is in charge of defining all the variables of patch types. It provides the retrieval and setting methods to obtain and modify the value of these variables. These methods are called in the Parcelle class. It is used to enhance the encapsulation of the system.
4) Cellule class: Cellule is also a French word standing for cell. This class is in charge of managing the biomass of one cell. It aggregates Compartment classes and Facies class. Constructor method of Cellule class defines 4 objects of these 4 compartment classes. Init_Cellule method initializes all the variables of cells and calls initialization methods of 4 compartment objects. Cut method computes the harvest biomass and the reserved biomass of one cell at the time of cut event finishing. Fertilization method changes the value of variable NI after fertilization event. Grazing method

computes the situation of this cell during the period of grazing. The elements which should be considered include:

- If this cell is grazed by animal, the height of the grass in this cell turns to the default lowest value.
- If this cell belongs to the dung area of dung buffer area, it cannot be grazed during the period of grazing. Shift NI status of cells influenced by dung after 7 days where water is non-limiting.

5) Compartment class: It is the super class of these 4 specific compartment classes. This class only contains the common variables and methods of these 4 specific compartments as the biomass, the height of the grass and the harvest biomass. Cut method, fertilization method and grazing method are generated in Compartment class because these 3 actions have not strong connection with the type of compartment.
6) Compartment_VV class: This class contains the necessary variables for computing the biomass of compartment green vegetation. It extends from Compartment class and there are some private methods to compute the intermediate values during the period of vegetation growth. These private methods are just called by Compartment_VV class and invisible to the other class. It is used to enhance the encapsulation of the system and keeps the safety of data. Init_VV method is called by Cellule class in the phase of initialization of Cellule class. MiseAJour class is used to update the variables of compartment green vegetation during the period of simulation.
7) Compartment_VS class: The variables for analyzing the biomass of compartment green reproductive are contained in this class. It extends from Compartment class and it is the same as Compartment_VV class that it contains the private method to compute the intermediate values for Compartment_VS class itself. Init_VS method is called by Cellule class in the phase of initialization of Cellule class. MiseAJour class is used to update the variables of compartment green reproduction during the period of simulation.
8) Compartment_RV class: This class contains the necessary variables for computing the biomass of compartment Dead vegetation. It extends from Compartment class and there are also some private methods to compute the

intermediate values during the period of vegetation growth. These private methods are just called by Compartment_RV class and invisible to the other classes. Init_RV method is called by Cellule class in the phase of initialization of Cellule class. MiseAJour class is used to update the variables of compartment dead vegetation during the period of simulation.

9) Compartment_RS class: It includes all the variables for computing the biomass of compartment Dead reproduction. It extends from Compartment class and bespoke methods are used to compute the intermediate values during the period of vegetation growth. These bespoke methods are just called by Compartment_RS class and invisible to the other class. Init_RS method is called by Cellule class in the phase of initialization of Cellule class. MiseAJour class is used to update the variables of compartment dead reproduction during the period of simulation.

Chart 4-6 illustrates calling sequence of one time of simulation. In fact, this sequence diagram simplifies the process of one simulation and it just lists the main methods during the period of simulation.

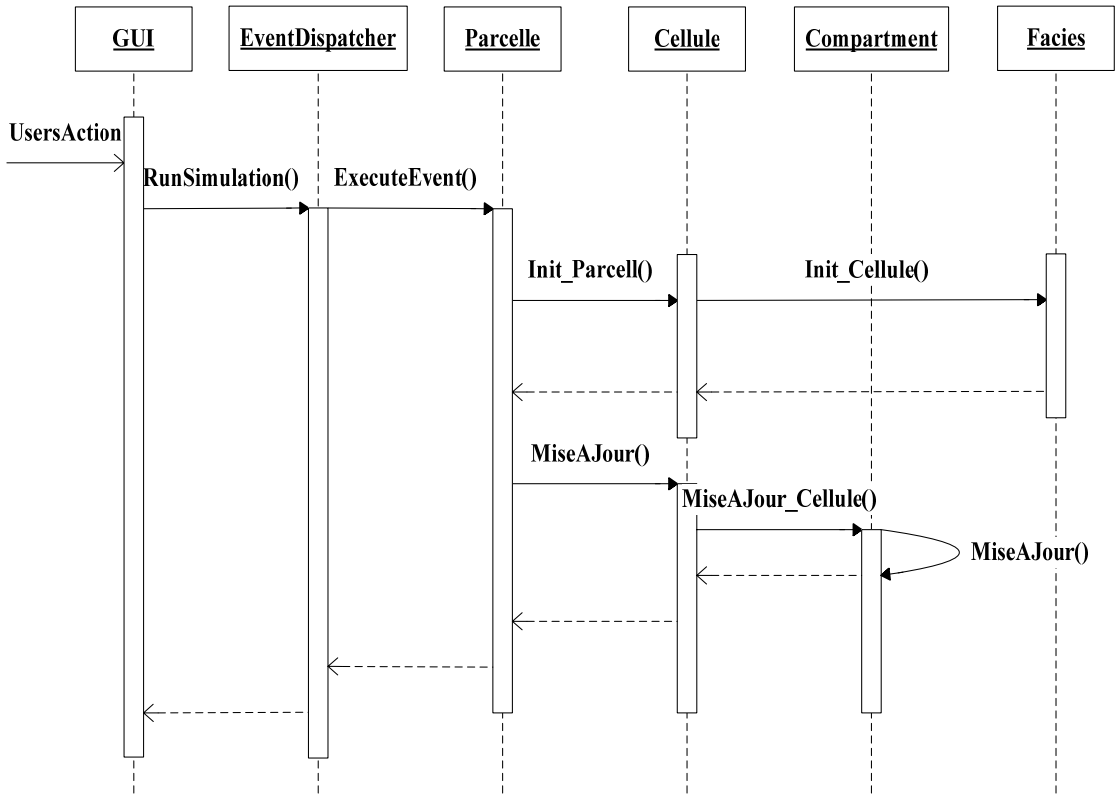


Chart 4-6 Sequence diagram of one vegetation simulation

First of all, users configure of the system by inputting the start time and the duration of the simulation, deciding the input file path, arranging the structure of the patch distribution, creating the events and configuring the events, determining the output file format… After configurations, users click the "Run" button to run the simulation and GUI executes "RunSimulation" method. This method will call the event dispatcher to execute the arranged event sequences. Event dispatcher will instantiate the Parcelle object and Parcelle object analyzes the biomass of each cell based on the event type. Cells are initialized by instantiating variables of the patch and compute the biomass of each compartment. Compartments compute the biomass and update their status then send their information back. At the end of simulation, the final data is displayed on the interface.

4.5 Key techniques

4.5.1 CSV file format

The comma-separated values (CSV) pseudo-file format is a set of file formats used to store tabular data in which numbers and text are stored in plain-text form that can be easily written and read in a text editor. It is a delimited data format that has fields/columns separated by the comma character and records/rows terminated by newlines. Fields that contains a special character (comma, newline, or double quote) must be enclosed in double quotes. If a line contains a single entry which is the empty string, it may be enclosed in double quotes. If a field's value contains a double quote character it is escaped by placing another double quote character next to it. The CSV file format does not require a specific character encoding, byte order, or line terminator format.

In this system, CSV file format is used in the input and output file. The library “Open CSV” is used to manipulate CSV file format. It provides a simply and efficient way to check data values and helps ecologist to analyze the situation of the pasture.

4.5.2 NetCDF file format

NetCDF (network Common Data Form) is a set of interfaces for array-oriented data access and a freely-distributed collection of data access libraries for C, Fortran, C++, Java, and other languages. The NetCDF libraries support a machine-independent format for representing scientific data. Together, the interfaces, libraries, and format support the creation, access, and sharing of scientific data. NetCDF has plenty of properties that can be used in the field of scientific computation and analysis.

1) A NetCDF file includes information about the data it contains.
2) A NetCDF file can be accessed by computers with different ways of storing integers, characters, and floating-point numbers.
3) A small subset of a large dataset may be accessed efficiently.
4) Data may be appended to a properly structured NetCDF file without copying the dataset or redefining its structure.
5) One writer and multiple readers may simultaneously access the same NetCDF file.

6) Access to all earlier forms of NetCDF data will be supported by current and future versions of the software.

In the system, NetCDF file format is used in the output file. The library from the group of NetCDF is used to manipulate NetCDF file format. NetCDF format provides a professional science way to compose data. Outputting NetCDF is capable to manage the output data more orderliness and facile to analyze and predict the future trend.

4.5.3 XML file configuration

Extensible Markup Language (XML) is a set of rules for encoding documents in machine-readable form. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards. Goals of XML emphasize simplicity, generality, and usability over the Internet. It is a textual data format with strong support via Unicode for the languages of the world. XML documents consist entirely of characters from the Unicode repertoire. Except for a small number of specifically excluded control characters, any character defined by Unicode may appear within the content of an XML document. The selection of characters that may appear within markup is somewhat more limited but still large. XML includes facilities for identifying the encoding of the Unicode characters that make up the document, and for expressing characters that, for one reason or another, cannot be used directly.

In this system, XML is used to store configuration data of one simulation. The configuration data includes input of users, site setting data, paths of input parameter files and format of output files.

4.5.4 Doxygen and Graphviz

Doxygen is a documentation system for C++, C, Java, Objective-C, Python, IDL (Corba and Microsoft flavors), VHDL, PHP, C#, and to some extent D. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and UNIX man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code. Doxygen can be configured to extract the code structure from undocumented source

files. This is very useful to quickly find your way in large source distributions. You can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.

Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains. The Graphviz layout programs take descriptions of graphs in a simple text language, and make diagrams in useful formats, such as images and SVG for web pages, PDF or Postscript for inclusion in other documents; or display in an interactive graph browser. Graphviz also supports GXL, an XML dialect. Graphviz has many useful features for concrete diagrams, such as options for colors, fonts, tabular node layouts, line styles, hyperlinks, Rolland custom shapes.

In this system, all the codes should be documented following JAVA documentation rules. Graphviz is integrated into Doxygen to generate not only documents but also charts to illustrate the structure of codes.

4.5.5 Vegetation growth model

This model is described in chapter 2 and is used in the vegetation part of the logic module. It is the foundation theory of the vegetation part. All the methods about computing biomass are based on this model. The computation is occurred in the classes of 4 compartments, updating variables of “Cellule” objects and in the “Parcelle” object, the final biomass will be computed corresponding to values of Cellule array. The Cellule array stands for the spatial structure of the grassland described the in chapter 2. Every element in this array represents each cell of the grassland. The variables of each elements accord to all the traits of each cell. The structure of vegetation part of the logic module is corresponding to the description of the vegetation growth model.

4.6 Brief summary

This chapter describes the system logic structure design, IO part specific design, GUI module specific design, Logic module specific design and related technology. In the system structure design part, the relationship of each module is illustrated and

described to keep the encapsulation of the system. IO part specific design narrates the main functions of IO part and the methods to fulfill the demands. GUI module specific design describes the class diagram of the GUI part. Each class is specialized and the design of main methods is discussed. Logic module specific design indicates 2 main parts of logic module and the relationship with each part. The class diagram of these 2 parts are showed and discussed specifically.

Chapter 5 System Implementation and Testing

5.1 The environment of system implementation

5.1.1 Technical condition

Programming language: JAVA

Modeling language: UML

Input file format: CSV (A kind of file format which illustrates data with the form of table), XML

Output file format: NetCDF (Network Common Data Form, a kind of scientific data format), CSV

Development tools: StarUML (A lightweight and free UML modeling tool), Eclipse, Berkeley Madonna(A modeling tool to illustrate the traits, the diversion conditions and the equations)

Version control: SVN

Documentation tools: Doxygen and Graphviz

5.1.2 Experimental condition

This system is implemented in JAVA by using Eclipse. So the experiment condition is focus on the Eclipse and its running condition. As well, the amount of data can be numerous, so the experiment condition is just able to deal with the common amount of data and expected least overflow in the memory.

CPU: Intel Core Duo T6500 @2.1GHz

RAM : 2.0GB

OS : Microsoft Windows 7

GPU : NVIDIA Geforce GT 130M @256MB

Hard disk: 250G

IDE: Eclipse

5.2 Implementation of modules

5.2.1 IO part implementation

From the description of design phase, the IO part is divided into 3 parts: input, output and XML configuration file handler. Input part is used to accumulate and record the data from input file. The format of input files is CSV so the input part is in charge of reading the file format CSV. There are many ways to read CSV file. In this system, a library named “Open CSV” is used and it is a jar package containing methods to read and write CSV file. For illustration more clearly, one method which deals with the input of patch type file is chosen to illustrate the main process of handling input files. Chart 5-1 shows the flow chart of input file method of inputting patch type.

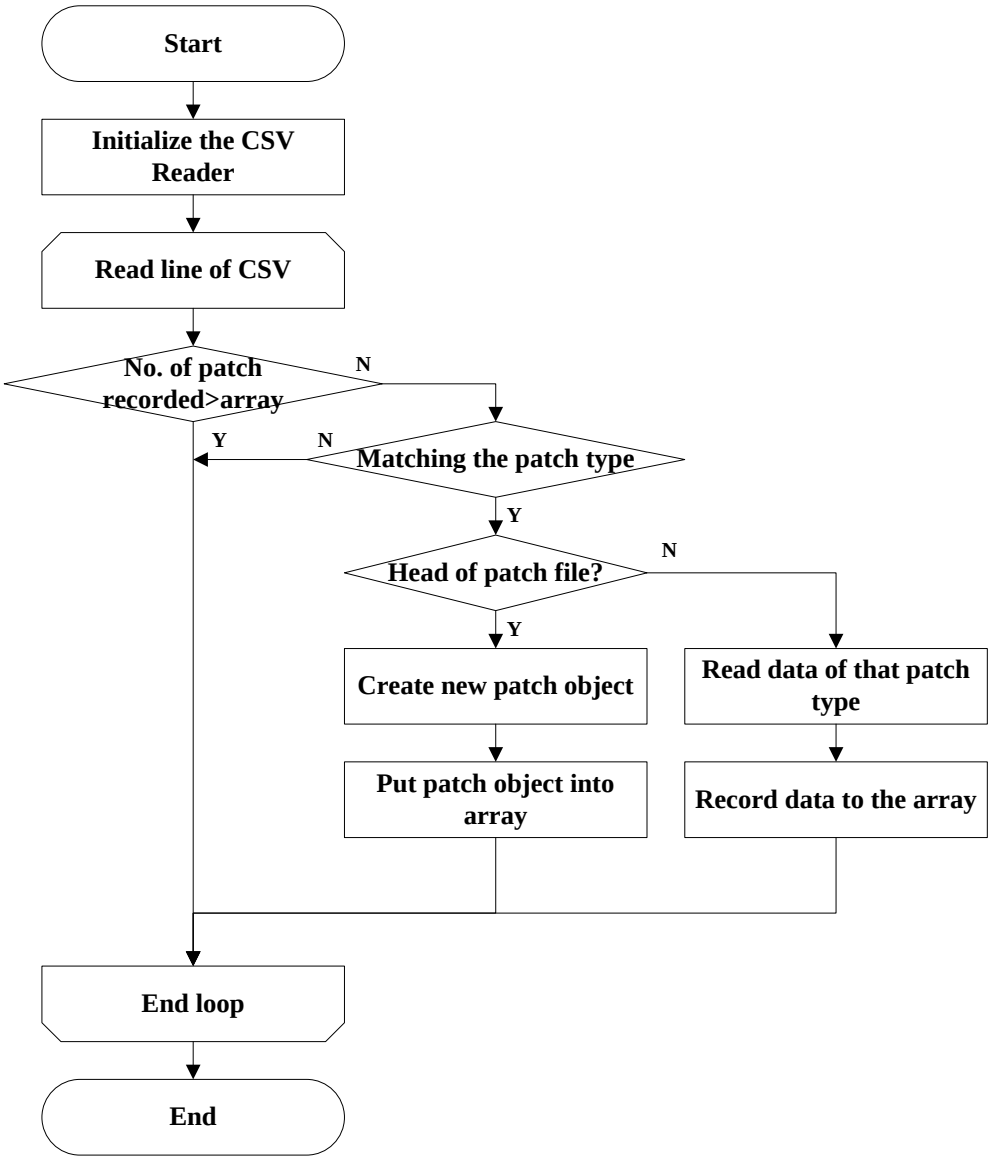


Chart 5-1 Input patch type flow chart

First of all, a CSV reader is created to open the target file and ready to read from this file. Every time, this CSV reader reads one line of CSV file. Because variables in CSV file are separated by some separators such as comma, variables in this line are clarified by separators. Then compare the content of the 1st variable with the name of variables in a patch enumeration. If there is a match, record the rest data into the patch type array. When the CSV reader reaches the end of the file, the record process stops and this CSV reader is unconstructed.

The second part of IO is outputting the data into CSV file or NetCDF file. The same as input, outputting to CSV uses “Open CSV” library. But for NetCDF file, un unique library provided by the “NetCDF” research group is used to output NetCDF

file. Because there are 2 kinds of outputting data which records the temporary data and the final data, 2 kinds of method are needed to deals with demands respectively.

1) Outputting temporary data: For CSV file, it is integrated into “Record per day” event which records the temporary data at the end of each day during the period of simulation. Parcelle class generates a CSV writer at beginning and sends it to the event execution method. This CSV writer outputs the temporary data into one CSV file and exists during the period of simulation. When the simulation ends, this CSV file is closed. For NetCDF file, all the temporary data is recorded in a special type of array which can be recognized by NetCDF writer. When the NetCDF writing operated, the content of this array is written into NetCDF file which is opened by NetCDF writer. Chart 5-2 shows the flow chart of writing NetCDF file.

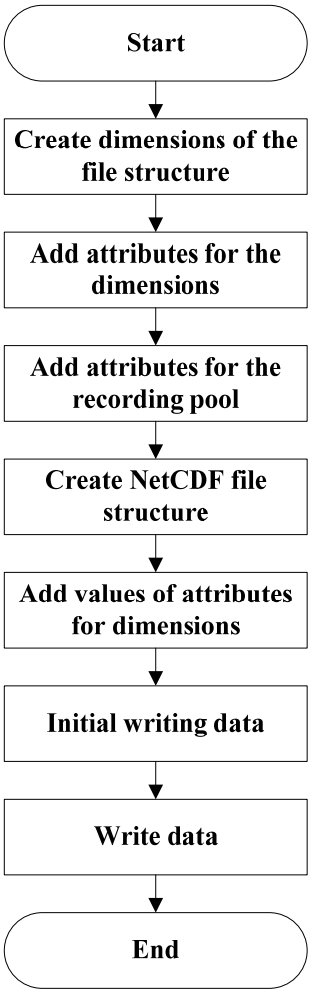


Chart 5-2 Writing NetCDF flow chart

For writing the NetCDF file, first of all, some dimensions should be created because of the structure of NetCDF file. Then add specific attributes to dimensions as well as the recording data pool. When these configurations are finished, the structure of NetCDF file is established. Then add values to attributes of dimensions. Before writing data, an original data should be provided to initialize the data pool. At last output data from the NetCDF format array to the files.

2) Outputting final data: Because final data is gotten at the end of the simulation, the writing operation just runs once to finish outputting data. For CSV file, it outputs the data of each cell as one line of CSV file. So the final amount of lines in the CSV file equals to the number of cells in line multiplying the number of cells in column. For NetCDF file, it records the data in an array then output the content of this array to the file.

The third part of IO is the XML configuration method. This part deals with the loading the XML file and outputting XML file. There are many ways to load and output XML file. In this system, DOM library is used to load and output XML file. This library is contained in the javax.xml package and it is the simplest way to operate XML file. Chart 5-3 shows the flow chart of creating XML configuration file.

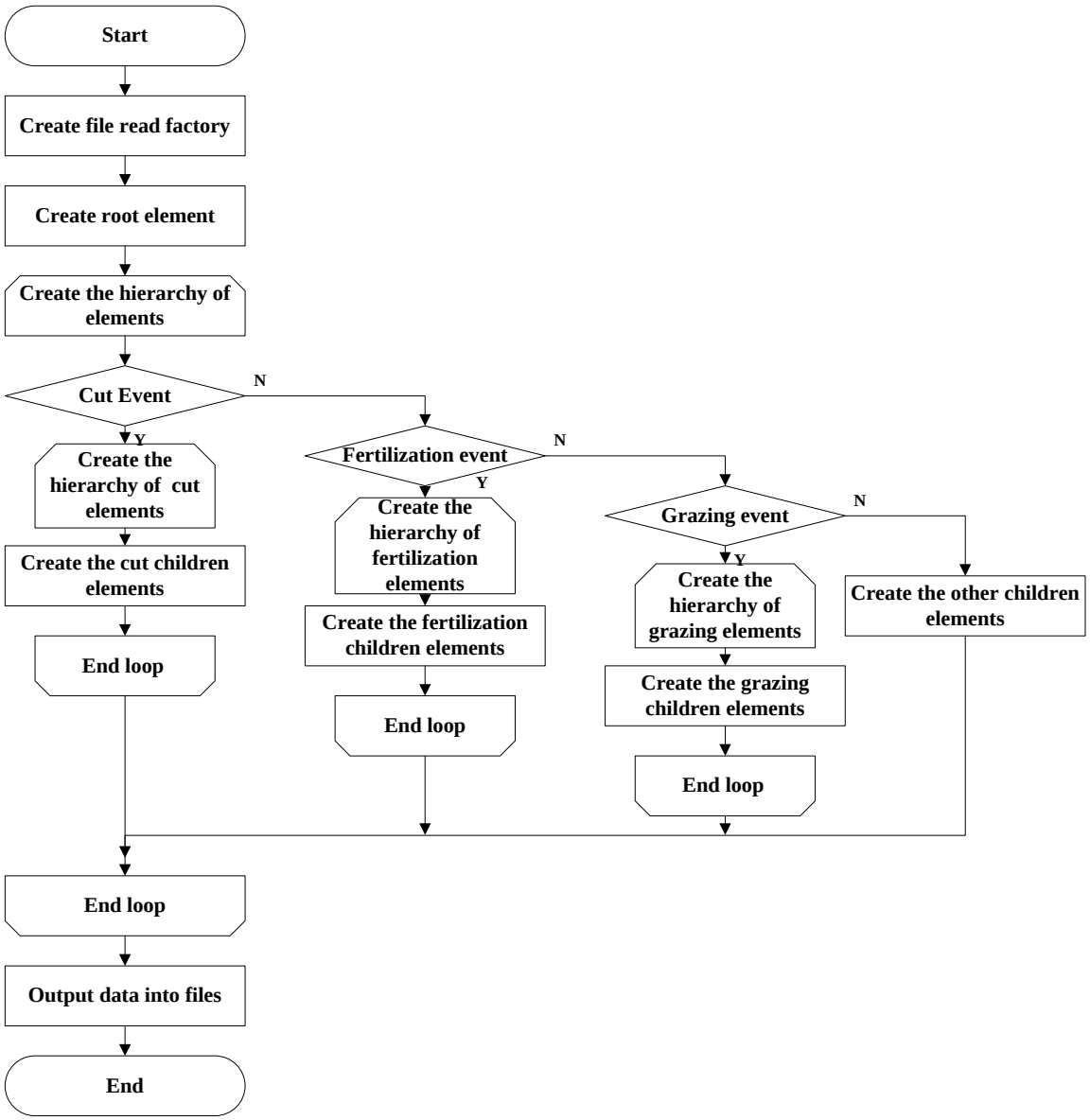


Chart 5-3 XML configuration flow chart

Using DOM to operate XML, first of all, a file reading factory is created to establish the foundation of the XML hierarchy structure. Then create the root element for the superior part. After that, the hierarchy is built according to the structure of data. Recursively create the children elements when there is no data to be added. At last output data into files.

5.2.2 GUI module implementation

The GUI module use JAVA swing tools to create the interface of the system. GUI class initializes the frame of the interface and MainPanel class creates the main

panel which contains all the tabs of the interface. The other panels are added to the MainPanel. The layout managers of all the panels are GridBagLayout. GridBagLayout is one of the most flexible — and complex — layout managers the Java platform provides. A GridBagLayout places components in a grid of rows and columns, allowing specified components to span multiple rows or columns. Not all rows necessarily have the same height. Similarly, not all columns necessarily have the same width. Essentially, GridBagLayout places components in rectangles (cells) in a grid, and then uses the components' preferred sizes to determine how big the cells should be. The main operation of the interface is adding components to panels such as text fields, labels, combo boxes…etc. Besides these adding component operation, GUI runs simulation when "run" button is clicked and sets patch distribution strategy.

For running simulation, Chart 5-4 shows run simulation flow chart of GUI module.

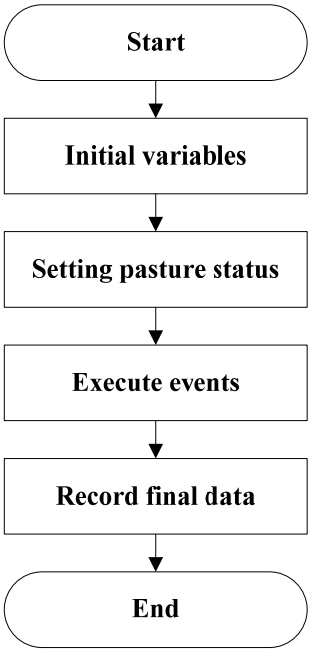


Chart 5-4 GUI module run simulation method flow chart

To run simulation, first of all, necessary variables are needed to be initialized. Then there are some setting operations to set pasture statues to ready for the simulation operation. After that, events are executed recursively and the data is computed. Finally, the final data is outputted into the file format which is selected before running simulation. There are the major codes for running simulation.

//Initialize variables

```
iDepart = Integer.parseInt(mainPanel.departField.getText());
iDuree = Integer.parseInt(mainPanel.dureeField.getText());
//Set pasture status
parcelle.setDepart(iDepart);
parcelle.setDuree(iDuree);
parcelle.InitParcelle(strParamFile, strFaciesFile, strEnvFile,
                    strConstantFile, strSiteFile, strCellFile, iDepart, iDuree);
parcelle.isOutputCSV(_outputCSV) ;
parcelle.isOutputNetcdf(_outputNetcdf) ;
//Run event dispatcher
eventDispatcher.excuteEvent(parcelle);
parcelle.recordFinalResult();
//Get result
setData(parcelle.getResult());
```

For setting the patch distribution strategy, the main points are selecting the method of strategy, operating that strategy and painting cells on the interface. Chart 5-5 shows the flow chart of set patch distribution strategy method in the GUI module. It takes aggregate method as an example.

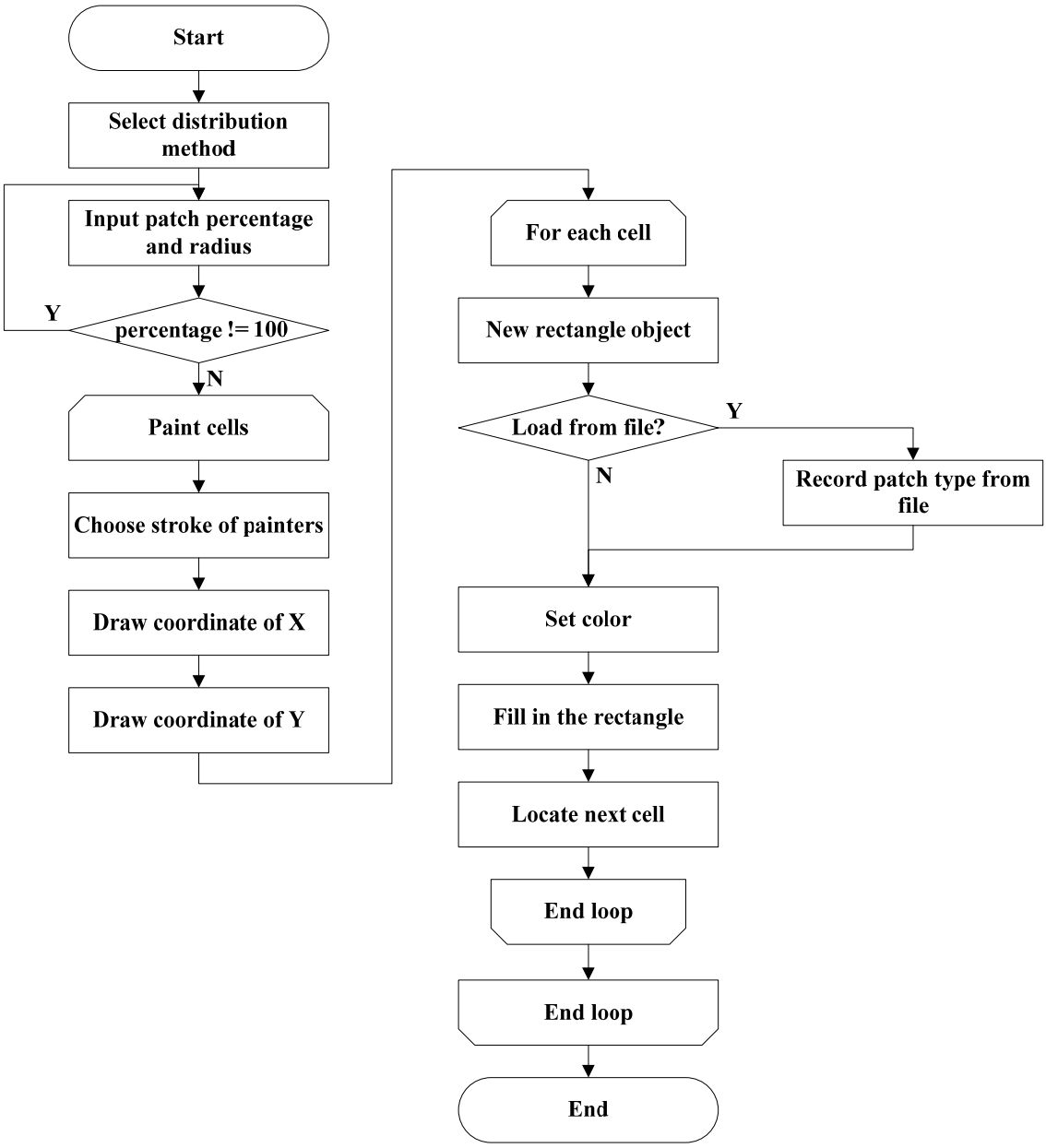


Chart 5-5 Set patch distribution strategy flow chart

Aggregate method is a kind of method to set patch distribution strategy. Chart 5-6 shows the flow chart of aggregate method which is used in the setting patch distribution strategy method.

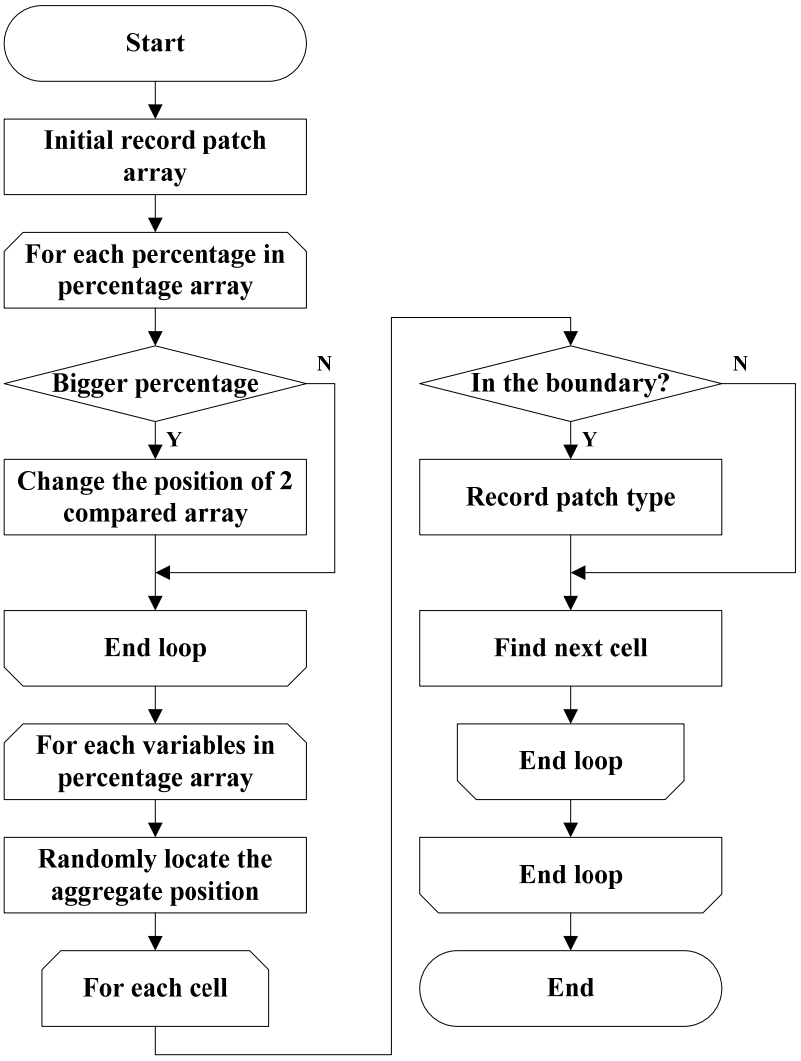


Chart 5-6 Aggregate method flow chart

To record the patch of each cell, there initialize an array with the same numbers of line and column with the pasture. According to the percentage of patch, the priority of each patch is set. High percentage means high priority. To set the priority of the patch, the percentage of patch is sorted from the maximum value to minimum value stored into an array. Then to get the random patch, a number from 1 to 100 is randomly selected. After that, it compares with the percentage stored in the priority array. This random number, depending on which percentage of the patch accords to its demand, will determine which patch type is selected. After setting the priority of patch, the aggregate method is used to generate the patch type. The major code of aggregate method is as following.

//Generate the number of cells aggregated according to the patch percentage

```
nbCellToSet =(int) (_iLine * _iColumn * _patchPercent[i]/100);
for (int l = 0; l < nbCellToSet; ) {
    //Randomly choose cell aggregated location
    line = (int)(Math.random()*_iLine+1);
    column = (int)(Math.random()*_iColumn+1);
    for(int j = line - _poissonR;j<=line+_poissonR;j++)
        for(int k = column-_poissonR;k<=column+_poissonR;k++)
        {
            if(j<0||k<0||j>_iLine-1||k>_iColumn-1||_patchRecord[j][k]!=0)
                continue;
            else {
                //Allocate the patch type to the cell according the priority
                //of patch
                _faciesType = _prioPatch[i];
                _patchRecord[j][k] = _faciesType;
                ++l;
            }
        }
}
```

5.2.3 Logic module implementation

Logic module implements event dispatcher and the vegetation growth. Event dispatcher is in charge of managing event sequences and vegetation part receives data from input files and computes the biomass according to the present event type.

1) Event dispatcher:

   Chart 5-7 shows the flow chart of event dispatcher method in the logic module.

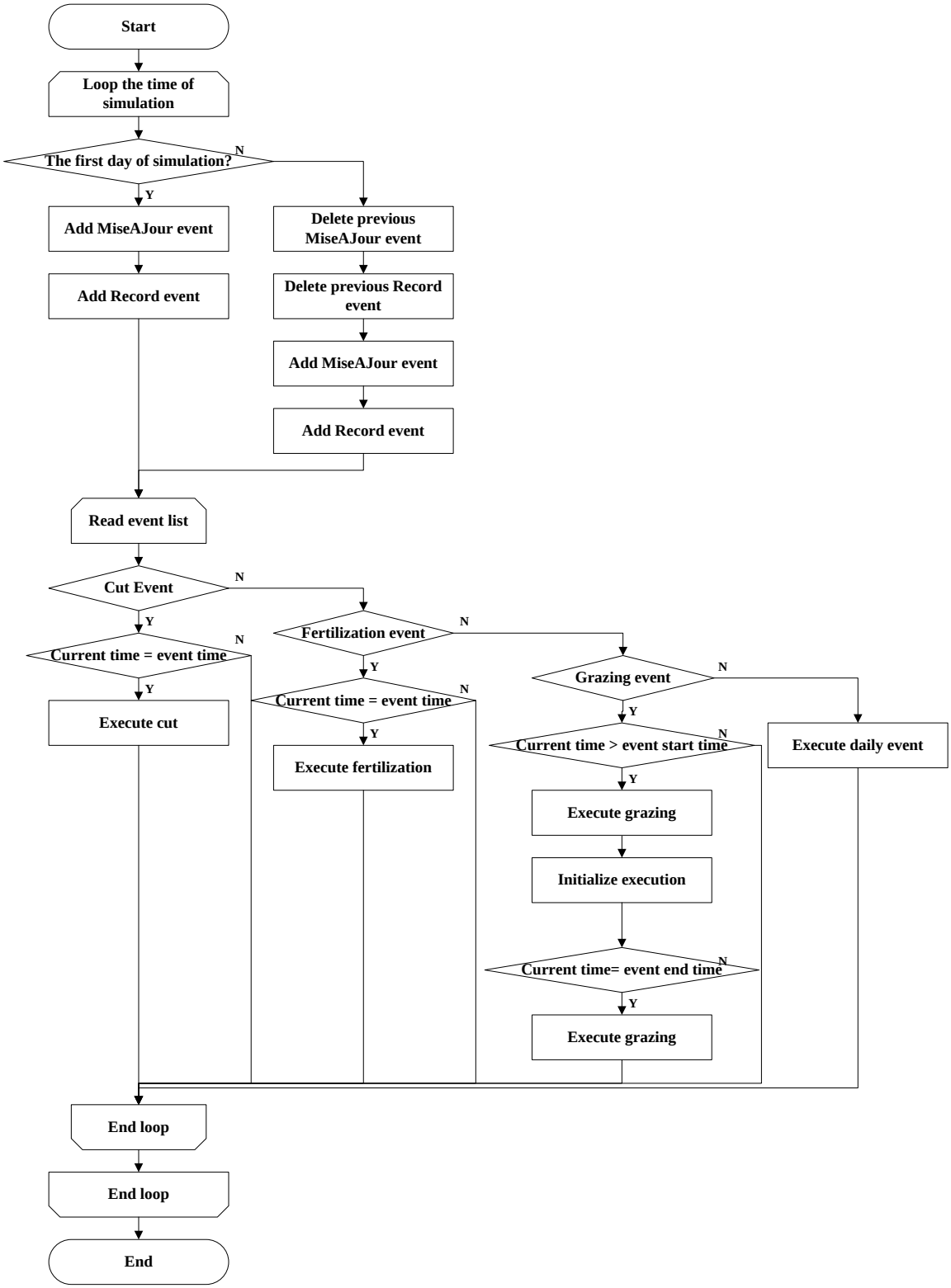


Chart 5-7 Event dispatcher flow chart

Event dispatcher gets the start time of simulation and the duration of simulation from the input file. Then it loops the event sequence to execute based on the time of

simulation. During each day's loop, "Mise A Jour" event (update vegetation) and "Record per Day" event are created and added into the event list. "Mise A Jour" event is created in the front of the event list and "Record per Day" event is added at the end of the event list. The codes are as following.

```
if( jour == _depart)
{
    addMiseAJourEvent("Mise A Jour", jour);
    addRecordPerDayEvent("Record Per Day", jour);
}
else {
    deleteEvent(0);
    deleteEvent(_eventList.size()-1);
    addMiseAJourEvent("Mise A Jour", jour);
    addRecordPerDayEvent("Record Per Day", jour);
}
```

Then elements of event list are checked if they accord to the condition of executing this event. If some element accords to the condition, it will be executed. And "Mise A Jour" event (update vegetation) and "Record per Day" event are executed for each day.

```
for(int i=0;i<_eventList.size();i++)
{
    if("Cut".equals(_eventList.get(i).getEventName()))
    {
        if(jour == _eventList.get(i).getCutStartTime())
        {
            _eventList.get(i).excuteEvent(parcelle);
        }
    }
    else if("Fertilization".equals(_eventList.get(i).getEventName()))
    {
        if(jour == _eventList.get(i).getFertStartTime())
        {
            _eventList.get(i).excuteEvent(parcelle);
        }
```

```
        }
        else if("Grazing".equals(_eventList.get(i).getEventName()))
        {
            if(jour< _eventList.get(i).getLoadAnimalEndTime()
                    &&jour>=_eventList.get(i).getLoadAnimalStartTime())
            {
                _eventList.get(i).excuteEvent(parcelle);
                _eventList.get(i).initExcute();
            }
            else if(jour == _eventList.get(i).getLoadAnimalEndTime())
            {
                _eventList.get(i).excuteEvent(parcelle);
            }
        }
        else {
            _eventList.get(i).excuteEvent(parcelle);
            }
        }
    }
```

Event dispatcher not only provides the method to execute events but also adding new events to the event list, deleting events from event list and modifying the event existed in the event list. All this operation is based on the classification of the event name. During the operation of event dispatcher, an event list is maintained to record the event sequence. This event list is an array list and the data format, which is contained in this array list, is “EventHandler” object. EventHandler is a child class of the super class event. It is able to generate an object for Event_Cut class, Event_Fertilization class, Event_Grazing class, Event_MiseAJour class and Event_RecordPerDay class. It composes these classes and is composed by EventDipatcher class. The purpose of existing of EventHandler class is to ensure the uniqueness of containing class in the array list. At the same time, it initializes the other event objects and provides methods necessary for the other events. It classifies different event objects by knowing event names. It provides methods for event dispatcher such as executing events and getting the value of specific variables of the event.

2) Vegetation part

Main class of vegetation part is “Parcelle”. It provides methods for the event dispatcher to execute different events, to initialize input variables and to output the final data. It generates a 2-dimention array to save the information of cells. The data format of this 2-dimention array is “Cellule” object and initialized in the “Parcelle” class. Each element stands for a cell object which contains all the information of a cell and computes the biomass of 4 compartments and total. In the “Cellule” class, objects of 4 compartments are initialized and join the common computation operations including computing LAI, fIN, AET, WR, W, CUT, ALLOC, PGRO, ENV and AN. As well, in the “Cellule” class, updating vegetation, cut, fertilization and grazing methods are provided to compute the biomass based on the event type. In the 4 compartment methods, the specific variables and constrains of computing the biomass for itself are created and initialized. For each compartment, AGE should be computed by its own constrains. For compartment_GV which stands for green vegetative, GRO and SEN of green vegetative are considered and computed. For compartment_GR which stands for green reproductive, GRO and ABS of green reproductive are considered and computed. For compartment_DV which stands for dead vegetative, GRO and SEN of dead vegetative are considered and computed. For compartment_DR which stands for dead reproductive, GRO and ABS of dead reproductive are considered and computed.

To accumulate the environment data, “Parcelle” creates an array for storing the environment information during the period of simulation. Because the number of environment data equals 365, the number of day in one year. Storing all the environment data wastes the memory space and no efficient.

To initialize the unique information of each cell, patch information is stored in the patch array which is created by “Parcelle” class and used by “Cellule” class. This array contains 4 patch types and is never changed during the simulation. Because there are enormous amounts of data, for easy-use and expansibility for the future, an enumeration inner class is created in “Facies” (patch) class to get the name of each variable of the patch type. The code of this enumeration class is as following.

```
public enum IndexVarFacies{
    Facies("Facies"),ST1("ST1"),ST2("ST2"),Incell("INcell"),Wcell("Wcell"),Hce
    ll("Hcell"),WHC("WHC"),minSEA("minSEA"),maxSEA("maxSEA"),W_VV(
```

```
    "W_VV"),alpha_PAR("alpha_PAR"),T0("T0"),T1("T1"),T2("T2"),beta_T("bet
    a_T"),b_IN("b_IN"),SLA("SLA"),LLS("LLS"),rho_VV("rho_VV"),gam_min(
    "gam_min"),gam_max("gam_max"),gammaVV("gammaVV"),W_RV("W_RV
    "),a_IN("a_IN"),max_fIN("max_fIN"),rho_RV("rho_RV"),W_VS("W_VS"),K
    _VS("K_VS"),Kl_VS("Kl_VS"),rho_VS("rho_VS"),W_RS("W_RS"),K_RS("
    K_RS"),Kl_RS("Kl_RS"),rho_RS("rho_RS"),init_AGE_VV("init_AGE_VV"),
    init_AGE_RV("init_AGE_RV"),init_AGE_VS("init_AGE_VS"),init_AGE_RS
    ("init_AGE_RS"),BD_VV("BD_VV"),BD_VS("BD_VS"),BD_RV("BD_RV")
    ,BD_RS("BD_RS");
    private final String name;
    IndexVarFacies(String name){
        this.name = name;
    }
    public String getName()
    {
        return this.name;
    }
}
```

5.3 Key Interfaces of the software system

1) General input tab: Chart 5-8 shows the general input tab of the system. This tab is in charge of inputting the start time and duration of the simulation.

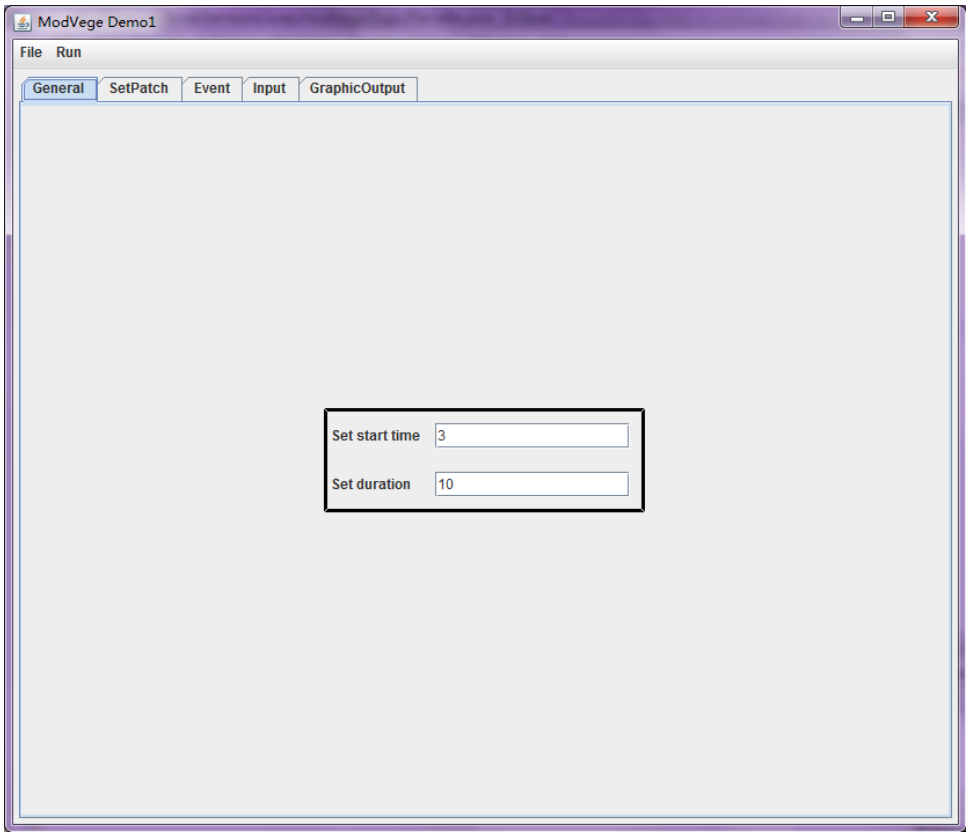


Chart 5-8 General input tab interface

2) Set patch tab: This tab is used to set patch distribution of the pasture. Chart 5-9 illustrates the set patch tab by choosing manual method. Chart 5-10 shows the set patch tab by choosing aggregate method.

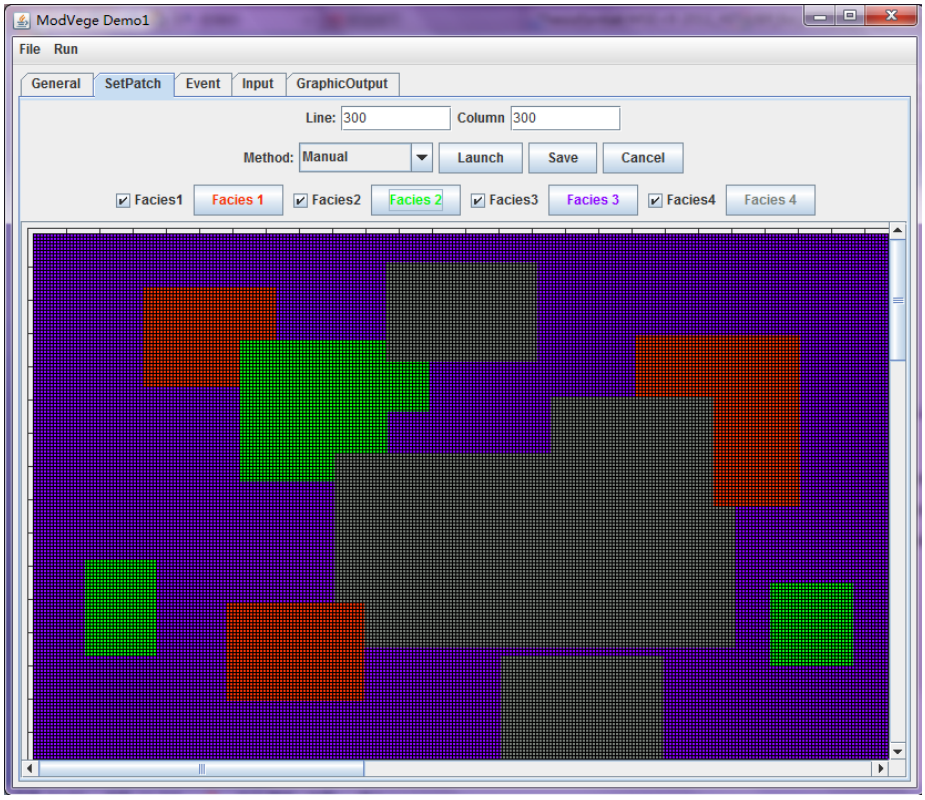


Chart 5-9 Set patch tab interface by choosing manual method

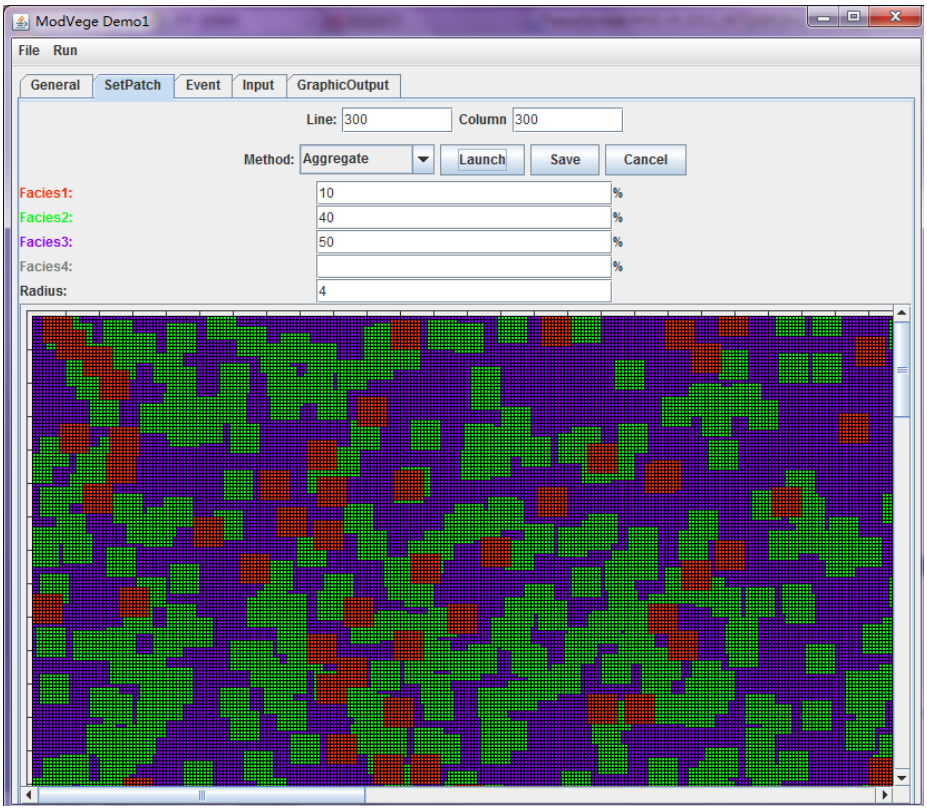


Chart 5-10 Set patch tab interface by choosing aggregate method

3) Event tab: This tab is used to manage the event sequence. Chart 5-11 displays the interface of event tab.

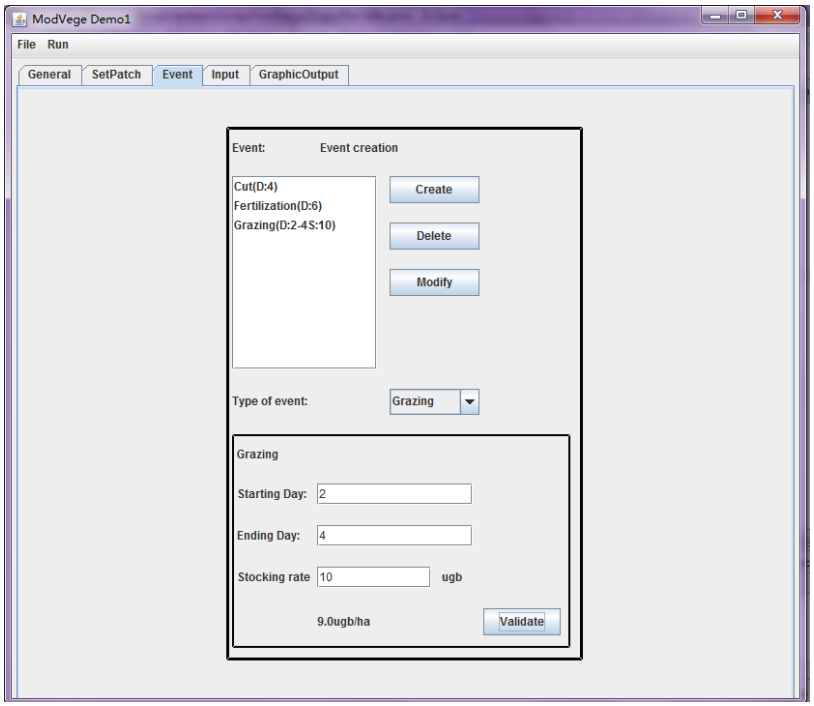


Chart 5-11 Event tab interface

4) Input tab: This tab is used as initializing paths of different types of input files. Chart 5-12 shows the input tab interface.

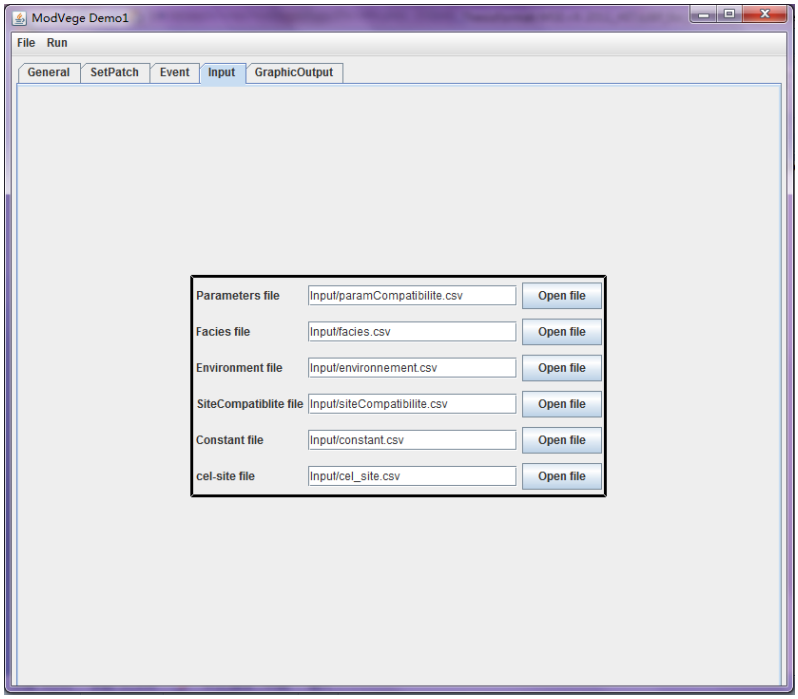


Chart 5-12 Input tab interface

5) Graphic output tab: This tab shows the result of each cell in a graphic way by choosing the compartment type. Chart 5-13 shows the graphic output tab interface.

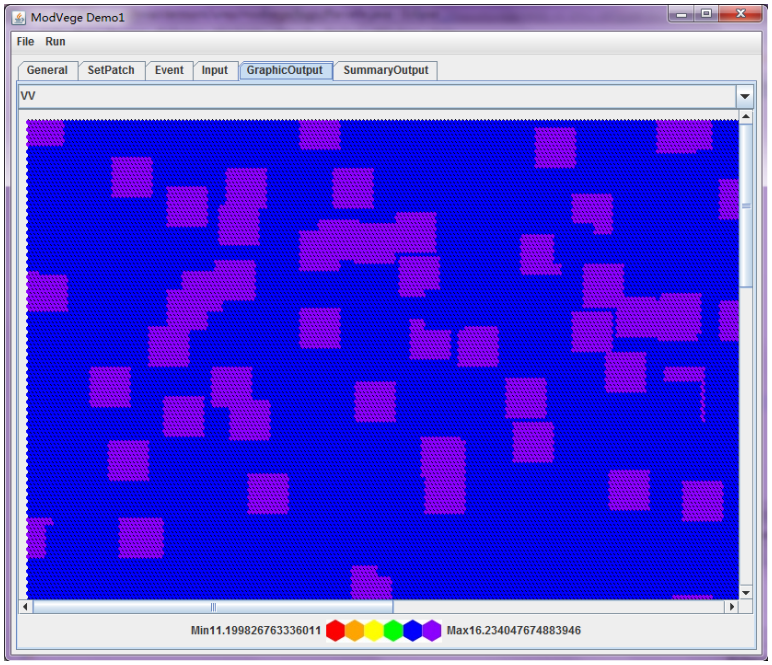
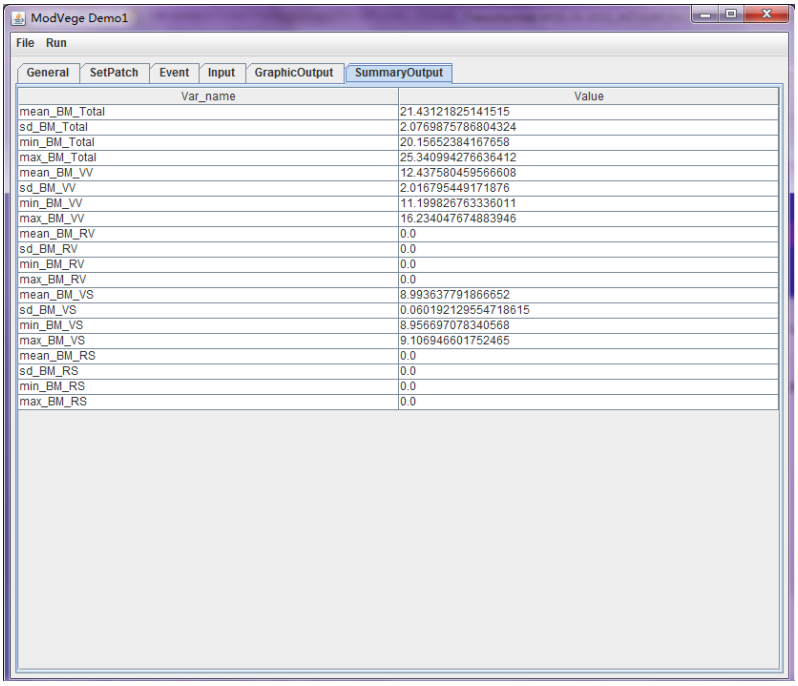


Chart 5-13 Graphic output tab interface

6) Summary output tab: This tab shows the summary of result data. Chart 5-14 shows the graphic summary output tab interface.



Var_name	Value
mean_BM_Total	21.43121825141515
sd_BM_Total	2.0769875786804324
min_BM_Total	20.156523841676 58
max_BM_Total	25.340994276636412
mean_BM_VV	12.437580459566608
sd_BM_VV	2.016795449171876
min_BM_VV	11.199826763336011
max_BM_VV	16.234047674883946
mean_BM_RV	0.0
sd_BM_RV	0.0
min_BM_RV	0.0
max_BM_RV	0.0
mean_BM_VS	8.993637791866652
sd_BM_VS	0.060192129554718615
min_BM_VS	8.956697078340568
max_BM_VS	9.106946601752465
mean_BM_RS	0.0
sd_BM_RS	0.0
min_BM_RS	0.0
max_BM_RS	0.0

Chart 5-14 Summary output tab interface

5.4 System Testing

5.4.1 System functional testing

System functional testing is in charge of testing if all the functions of the system run in a good condition without any bug and from the exact input getting the correct output. Table 5-1 shows criteria points of functional testing.

Table 5-1 Criteria points of functional testing

Module	Testing points	Description	Result
GUI	Load XML file	Load XML file and check if the general input text field, input file text filed and event list turns to the data stored in the XML file. Check if the set patch strategy tab shows the graphic display of the patch distribution, combo box of patch distribution method and the number of line as well as the number of column.	General input text field, input file text filed and event list shows the same result of XML file. Patch distribution strategy tab shows the graphic display same as the data in the “Cel-Site” file. Combo box shows “Choosing from file”. Line and column text fields show the number of line and column same as the parameter file.
	Load input file manually	Load input file from clicking “Open file” to add a file path.	Click “Open file” button to open a file selection dialogue to choose a file. After choosing the file, the path is added into the text field of input file.
	Load patch strategy from file	Choose patch distribution method “Choose from file”. Select the “Cel-site” file then load it. The graphic part shows the patch distribution strategy.	After loading file, patch distribution strategy is shown in the graphic part and it is as same as “Cel-site” file.

Table 5-1 (continued)

Module	Testing points	Description	Result
GUI	Set patch strategy using manual method	Choose patch strategy method “Manual” and input the number of line and column to generate the initial pasture. Then choose patch type and draw the patch area by using pulling mouse.	After choosing method and inputting line and column, the initial pasture is successfully created. Then choose the patch type to draw the patch area. Finally save the patch distribution in the file. The content of file is the same to the graph on the interface.
	Set patch strategy using random method	Choose patch strategy method “Random” and input the percentage of each patch type. Then launching the result, a graph should be shown on the interface.	After choosing method, inputting the percentage of patch type, and launching the result, a graph is shown on the interface which has the same distribution of input patch percentage. Finally save the patch distribution in the file. The content of file is the same to the graph on the interface.
	Set patch strategy using aggregate method	Choose patch strategy method “Aggregate” and input the percentage of each patch type as well as the radius. Then launching the result, a graph should be shown on the interface.	After choosing method, inputting the percentage of patch type, inputting the radius and launching the result, a graph is shown on the interface which has the same distribution of input patch percentage. Finally save the patch distribution in the file. The content of file is the same to the graph.

Table 5-1 (continued)

Module	Testing points	Description	Result
GUI	Save XML file	Save the current configuration to XML file	Click save button to save the configuration into default XML file or click "save as" button to save it into a new file. After saving, check the content of XML file is the same to the current configuration.
Logic	Create event	Create events by inputting the information of that event.	Click "create" and choose a event type. Input the event information, for example, if the event type is "Cut", enter the start time of cut and default height. Click "validate" to create a new event. That event is shown in the event list.
	Delete event	Delete an event from event list.	Choosing an event existed in the event list, click delete. That event disappears from the event list.
	Modify event	Modify an event from event list.	Click "Modify" button to enter modify state. Choose an event existed in the event list. The information of that event is showed under the event list. Change values and click "validate". The event information is modified in the event list.

Because this system is an analyzed system, it is needed to test whether it could output the right result. For testing the correctness of input and output, test cases should be established. Due to the enormous variables and the page limitation of this thesis, one group of experiment data is shown as following. Table 5-2 shows the values of input environment variables. Table 5-3 shows the values of input patch variables. Chart 5-15 shows the distribution of patch type.

Start time: 3rd day.

Duration: 20 days.

Event occurrence: Fertilization occurs in 4th day. Grazing occurs from 5th day to 10th day.

Scale of pasture: 300 X 300

Table 5-2 Environment variable values

Day	Temperature	PARi	PP	PET
3	6.5	2.2656	3	0.5
4	5	2.328	2	0.9
5	4.4	1.6752	16.4	1
6	6.2	1.7328	0.4	0.3
7	7	2.1744	0.2	0.7
8	6.3	2.8176	0	0.2
9	7.4	2.8224	0	0.3
10	8.3	2.8608	0	0.7
11	7.1	2.3376	0	0.5
12	6.8	1.4688	0	0.8
13	7.1	1.2096	6	1
14	3.5	2.1216	0.6	0.5
15	2.7	0.8112	1.4	0.4
16	3.4	0.96	11.8	0.5
17	2.9	2.8896	0	0.5
18	3	1.3968	24.2	2
19	1.8	1.8528	4.6	0.2
20	-2.8	1.0752	0.2	0.4

Table 5-3 Patch variable values

Facies	1	2	3	4
ST1	600	500	400	500
ST2	1200	1200	1100	1200
INcell	0.7	0.7	0.7	0.7
Wcell	25	25	25	25
Hcell	0.0684	0.0684	0.0684	0.0684
WHC	200	100	150	200
minSEA	0.8	0.8	0.8	0.8
maxSEA	1.2	1.2	1.2	1.2
W_VV	14.5	12	10	11
alpha_PAR	0.044	0.044	0.044	0.044
T0	5	5	5	5
T1	10	10	10	10
T2	20	20	20	20
beta_T	0.05	0.05	0.05	0.05
b_IN	0.25	0.25	0.25	0.25
SLA	0.025	0.025	0.025	0.025
LLS	500	500	500	500
rho_VV	850	850	850	850
gam_min	0.4	0.4	0.4	0.4
gam_max	0.7	0.7	0.7	0.7
gammaVV	0.68	0.68	0.68	0.68
W_RV	0	0	0	0
a_IN	0.2	0.2	0.2	0.2
max_fIN	0.7	0.7	0.7	0.7
rho_RV	300	300	300	300
W_VS	10.5	10.5	10.5	10.5
K_VS	0.002	0.002	0.002	0.002
Kl_VS	0.001	0.001	0.001	0.001
rho_VS	500	500	500	500
W_RS	0	0	0	0
K_RS	0.001	0.001	0.001	0.001

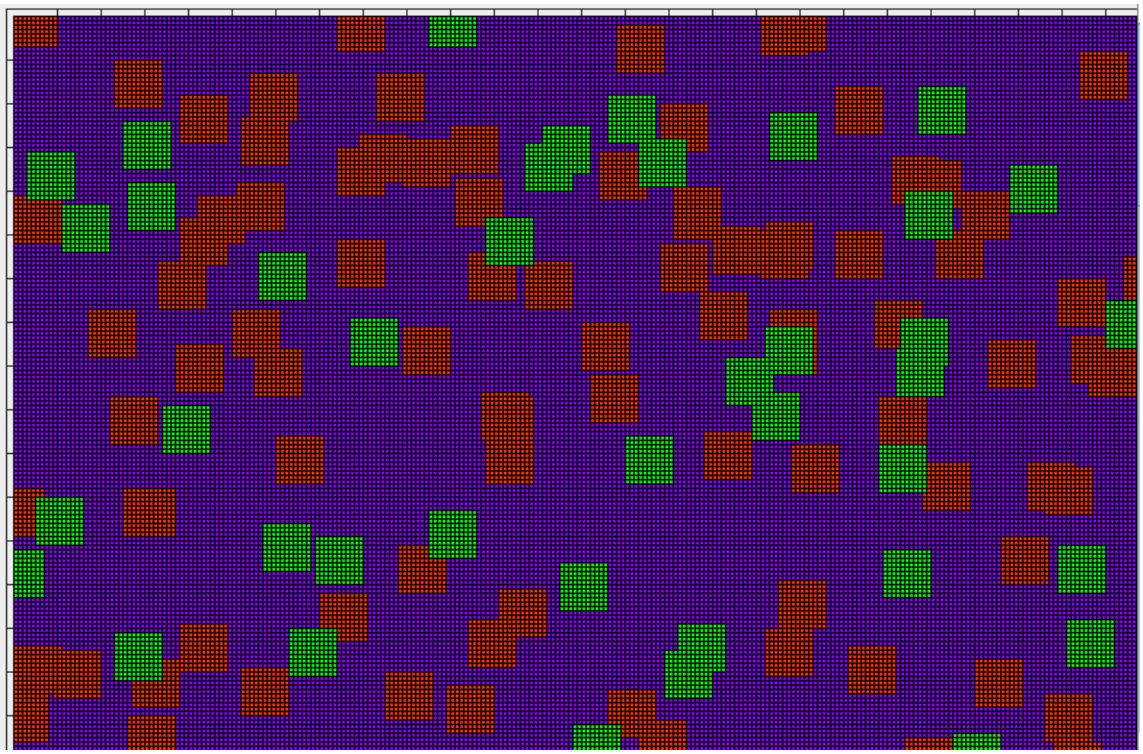


Chart 5-15 Patch distribution

For measuring if the final data is right, Berkeley Madonna is used to get the curve chart of the final data. Chart 5-16 shows the result in the output CSV file and chart 5-17 shows the curve chart by using Berkeley Madonna.

day	mean_BM_T	sd_BM_Tot	min_BM_T	max_BM_T	mean_BM_VV	sd_BM_VV	min_BM_VV	max_BM_VV	mean_BM_R	sd_BM_RV	min_BM_RV	max_BM_RV
3	24.99328	0	24.99328	24.99328	15.65453473	0	14.5	14.65453	0	0	0	0
4	24.8382	0	24.8382	24.8382	15.57540025	0	14.5754	14.5754	0	0	0	0
5	24.70273	0	24.70273	24.70273	15.48794784	0	14.48795	14.48795	0	0	0	0
6	24.63381	0	24.63381	24.63381	15.58293997	0	14.48795	14.58294	0	0	0	0
7	24.6776	0	24.63381	24.6776	14.77654774	0	14.58294	14.77655	0	0	0	0
8	24.70795	0	24.6776	24.70795	14.97630296	0	14.77655	14.9763	0	0	0	0
9	24.89951	0	24.70795	24.89951	15.35693988	0	14.9763	15.35694	0	0	0	0
10	25.24429	0	24.89951	25.24429	15.87483309	0	15.35694	15.87483	0	0	0	0
11	25.35768	0	25.24429	25.35768	16.14017217	0	15.87483	16.14017	0	0	0	0
12	25.34099	0	25.34099	25.34099	16.23404767	0	16.14017	16.23405	0	0	0	0
13	25.31259	0	25.31259	25.31259	16.30222179	0	16.23405	16.30222	0	0	0	0
14	25.21799	0	25.21799	25.21799	16.25331513	0	16.25332	16.25332	0	0	0	0
15	25.14537	0	25.14537	25.14537	16.21430717	0	16.21431	16.21431	0	0	0	0
16	25.05428	0	25.05428	25.05428	16.16566425	0	16.16566	16.16566	0	0	0	0
17	24.97694	0	24.97694	24.97694	16.11716726	0	16.11717	16.11717	0	0	0	0
18	24.89721	0	24.89721	24.89721	15.92376125	0	15.92376	15.92376	0	0	0	0
19	24.84875	0	24.84875	24.84875	15.90465274	0	15.90465	15.90465	0	0	0	0
20	24.81058	0	24.81058	24.81058	15.86648157	0	15.86648	15.86648	0	0	0	0
21	24.7725	0	24.7725	24.7725	15.82840201	0	15.8284	15.8284	0	0	0	0
22	24.7535	0	24.7535	24.7535	15.80940793	0	15.80941	15.80941	0	0	0	0

Chart 5-16 Result in the output CSV file

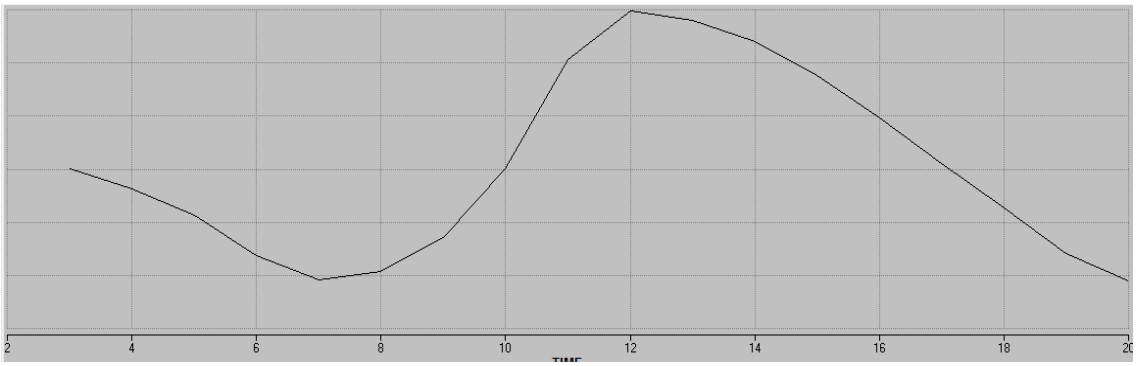


Chart 5-17 Curve chart of Berkeley Madonna

The result data shows the same trend of the curve chart of Berkeley Madonna. So the result is correct.

5.4.2 Pressure testing

Because this system is aiming to enormous amount of data, the pressure test is necessary to measure the stability, run time, throughput and memory allocation of the system. There are some test cases to examine the system for pressure testing. Because writing NetCDF file costs long time and large memory allocation, each group of test case is designed to 2 kinds with NetCDF file output and without NetCDF file output.

1) Test 1-1

   Scale of pasture: 500 X 500

   Duration of simulation: 200 day

   Output NetCDF file format: No

   Run condition of the system: Successfully executed.

   Run time: 50 seconds.

   Test 1-2

   Scale of pasture: 500 X 500

   Duration of simulation: 200 day

   Output NetCDF file format: Yes

   Run condition of the system: Successfully executed.

   Run time: 70 seconds.

2) Test 2-1

   Scale of pasture: 1000 X 1000

   Duration of simulation: 200 day

   Output NetCDF file format: No

Run condition of the system: Successfully executed.

Run time: 150 seconds

Test 2-2

Scale of pasture: 1000 X 1000

Duration of simulation: 200 day

Output NetCDF file format: No

Run condition of the system: Out of memory.

To conclude the pressure testing, the limitation of this system deals with 1000 X 1000 scales of pasture in 200 days without outputting NetCDF file. Outputting NetCDF file will increase the overhead of the system not only the running time but also the memory allocation.

5.5 Brief summary

This chapter describes the implementation of the system according to the design method showed in Chapter 4. It narrates the implementation of each module especially for the major part of the module. The interface of the system is shown and at last the test part is discussed. At present, the system accords the demands of all the testing points. Because of the limitation of the length of the article, just one test case is selected to show the process of testing the correctness of the final result. At last, the pressure testing is discussed to test the tolerance, reaction speed and memory allocation of the system.

Conclusion

This thesis implements a system to analyze the spatial structure and biomass of a pasture. This system provides a model to do the analysis and a Graphical User Interface which is interfaced with the model. The result of pasture structure and biomass dynamic analyzing system is as following.

1) This system implements the pasture analyzing model which utilizes the program to investigate the ecological knowledge by transferring the biology method to computer science language.
2) The model deals with numerous variables and constraints in an efficient way and simulates the conditions of pastures not only the measurable traits but also the physical structure in a logical way.
3) A GUI which is programmed in JAVA is established in this system. This GUI provides an easy tool for ecologist to directly control species traits, to manage events occurred during the simulation and to get the analyzed result in multiple ways.
4) This system provides the output file in two kinds of format for different objectives. CSV format is used frequently and easily for statistics by ecologist. NetCDF format is specialized in the domain of science analysis and is able to handle numerous amounts of data in a structured way.

There are also some defects of this system. In the original imagination, the analyzing model would be integrated into CAPSIS (Computer-Aided Projection of Strategies In Silviculture) platform to gain a better result display (3D distribution graph chart) and a better scalability. But the structure of CAPSIS platform is quite complex and it is difficult to adapt it to new needs in order to use the new model in a short time. From the result of pressure testing, the limitation and the memory allocation of this system is not ideal enough. It is needed to think further to optimize the system. The future tasks of this system are optimizing the memory allocation, increasing the computation efficiency and managing to integrate the model into CAPSIS platform.

References

[1] M. Jouven, P. Carrere and R. Baumont. Model predicting dynamics of biomass, structure and digestibility of herbage in managed permanent pastures [J]. Grass and Forage Science. 2006, 61, 112–124.

[2] Al Haj Khaled R., Duru M., Theau J.-P., Plantureux S. and Cruz P. Variation in leaf traits through seasons and N-availability levels and its consequences for ranking grassland species [J]. Journal of Vegetation Science. 2005, 16, 391–398.

[3] Baumont R., Ginane C., Garcia F. and Carrere P. How herbivores optimize diet quality and intake in heterogeneous pastures, and consequences on vegetation dynamics [R]. Pastoral systems in Marginal Environments. Satellite workshop of the 20th International Grassland Congress. 2005, 39–50.

[4] Bausenwein U., Millard P. and Raven J.A. Remobilized old-leaf nitrogen predominates for spring growth in two temperate grasses [J]. New Phytologist. 2005, 152, 283–290.

[5] Bullock J.M., Franklin J., Stevenson M.J., Silvertown J., Coulson S.J., Gregory S.J. and Tofts R. A plant trait analysis of responses to grazing in a long-term experiment [J]. Journal of Applied Ecology. 2001, 38, 253–267.

[6] Carrere P., Force C., Soussana, J.F., Louault F., Dumont B. and Baumont R. Design of a spatial model of perennial grassland grazed by a herd of ruminants: the vegetation sub-model [D]. Grassland Science in Europe. 2001, 7, 282–283.

[7] Coleno F.C. and Duru M. A model to find and test decision rules for turnout date and grazing area allocation for a dairy cow system in spring [D]. Agricultural Systems. 1999, 61, 151–164.

[8] Dıaz S., Cabido M. and Casanoves F. Plant functional traits and environmental filters at a regional scale [J]. Journal of Vegetation Science. 1998, 9, 113–122.

[9] Dıaz S., Noy-Meir I. and Cabido M. Can grazing response of herbaceous plants be predicted from simple vegetative traits [J]. Journal of Applied Ecology. 2001, 38, 1–12.

[10]Leon Coromoto, Miranda Gara. A Parallel Plugin-based Framework for Multi-Objective Optimazation[J]. International Journal of Artificial Intelligence Tools, 2009, 18（4）: 72-79.

[11]Dolling P.J., Robertson M.J., Asseng S., Ward P.R. and Latta R.A. Simulating lucerne growth and water use on diverse soil types in a Mediterranean-type environment [J]. Australian Journal of Agricultural Research. 2005, 56, 503–515.

[12]Duru M. and Ducrocq H. Growth and senescence of the successive grass leaves on a tiller. Ontogenic development and effect of temperature[D]. Annals of Botany, 2000, 85, 635–643.

[13]Duru M., Delprat V., Fabre C. and Feuillerac E. Effect of nitrogen fertilizer supply and winter cutting on morphological composition and herbage digestibility of Dactylis glomerata L. swards in spring [J]. Journal of the Science of Food and Agriculture. 2000, 80, 33–42.

[14]Proisy C., Souza Filho P., Prost M. T., Fromard F., Mendes A. C., de Coligny F. Monitoring the dynamic of the Amazon coast (Pará, Brasil and French Guiana) using a common methodology based on a spatial analysis coupled to a simulation Tool [D]. In Proceeding of the Mangrove 2003 conference. 2003, 20-24 May,

[15]Garcia F., Carre`re P., Soussana J.F. and Baumont R. The ability of sheep at different stocking rates to maintain the quality and quantity of their diet during the grazing season [J]. Journal of Agricultural Science, Cambridge. 2003, 140, 113–124.

[16]Garcia F., Carre`re P., Soussana J.F. and Baumont R. How do severity and frequency of grazing affect sward characteristics and the choices of sheep during the grazing season [D].Grass and Forage Science. 2003, 58, 138–150.

[17]Garnier E., Laurent G., Bellmann A., Debain S., Berthelier P., Ducout B., Roumet C. and Navas M.-L. Consistency of species ranking based on functional leaf traits [J]. New Phytologist, 2001, 152, 69–83.

[18]Ginane C., Petit M. and D'Hour P. How do grazing heifers choose between maturing reproductive and tall or short vegetative swards [J]. Applied Animal Behaviour Science. 2003, 83, 15–27.

[19]Groot J.C.J. and Lantinga E.A. (2004) An object-oriented model of the morphological development and digestibility of perennial ryegrass [D]. Ecological Modelling. 2004, 117, 297–312.

[20]Herrmann A. and Schachtel G.A. OSYAQ, an organspecific growth model for forage grasses [J]. Grass and Forage Science.2001, 56, 268–284.

[21]Lavorel S. and Garnier E. Predicting changes in community composition and ecosystem functioning from plant traits: revising the Holy Grail [D]. Functional Ecology. 2002, 16, 545–556.

[22]Louault F., Pillar V.D., Aufre`re J., Garnier E. and Soussana J.-F. Plant traits and functional types in response to reduced disturbance in semi-natural grassland [J]. Journal of Vegetation Science. 2005, 16, 151–160.

[23]McCall D. G. and Bishop-Hurley G.J. A pasture growth model for use in a whole-farm dairy production model [J]. Agricultural Systems. 2003, 76, 1183–1205.

[24]Milne J.A. and Sibbald A.R. (1998) Modelling of grazing systems at the farm level. Annales de Zootechnie [J]. 1998, 47, 407–417.

[25]Lawrence A. Cunningham. "Language, Deals and Standards: The Future of XML Contracts" [D]. Washington University Law Review. 2005, 49. 308-347.

[26]L. Duan, Y. Zhang, B. Li, and L. Peng. Universal Rules Guided Design Parameter Selection for Soft Error Resilient Processors [D]. Proceedings of the International Symposium on Performance Analysis of Systems and Software(ISPASS). 2011, 16, 201-222.

[27]M. Abramovici, M. A. Breuer, A. D. Friedman. Digital Systems Testing and

Testable Design [C]. Computer Science Press, New York, 1990.

[28]S. K. Abd-El-Hafiz, V. R. Basili, and G. Caldiera, Towards Automated Support for Extraction of Reusable Components [D]. Proc. IEEE Conference of Software Maintenance. 1991, 212-219.

[29]J. Benkoski and R. B. Stewart, TATOO: An Industrial Timing Analyzer with False Path Elimination and Test Pattern Generation [D]. Proc. European Conf. on Design Automation. 1991, 256-260.

[30]Arango, G., Williams, G. and Iscoe, N. Domain Modeling for Software Engineering [D]. Proceedings of the ICSE 1991 Domain Modeling Workshop. Austin Laboratory for Software Engineering and Computer Science. 1991, 1-4.

[31]Bobrow, D. and Stefik, M. Object Oriented Programming: Themes and Variations [J]. AI Magazine. 2006, 6(4), 40-62.

[32]Boehm, B. A Spiral Model of Software Development and Enhancement [D]. IEEE Computer. 1988, 21(5), 61-72.

[33]Booch, G. Object-Oriented Development [D]. IEEE Transactions on Software Engineering. 2001, 12(2), 211-221.

[34]Fischer, G., Grudin, J., Lemke, A., McCall, R., Ostwald, J., Reeves, B., and Shipman, F. Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments [J]. Human-Computer Interaction. 2005, 7(3), 281-314.

[35]Kant, E., Daube, F., MacGregor, W. and Wald, J. Scientific Programming by Automated Synthesis [C]. Automating Software Design. AAAI Press. 1991

[36]Selfridge, P., and Terveen, L., and Long, M. Managing Design Knowledge to Provide Assistance to Large-Scale Software Development [R]. Proceedings of the Seventh Knowledge-Based Software Engineering Conference. IEEE Computer Society Press. 1992

[37]Thornton B. and Bausenwein U. Seasonal protease activity in storage tissue of the deciduous grass Molinia cerulean [J]. New Phytologist. 2000,146, 75–81

[38]Adler PB, Lauenroth WK (2000) Livestock exclusion increases the spatial

heterogeneity of vegetation in the shortgrass steppe, Colorado [J]. Appl Veg Sci. 2000, 3, 213–222.

[39]Johnson I.R. and Parsons A.J. Use of a model to analyse the effects of continuous grazing management on seasonal patterns of grass production [J]. Grass and Forage Science. 1985, 40, 449–458.

[40]Lavorel S. and Garnier E. (1997) Plant functional classifications: from general groups to specific groups based on response to disturbance [J]. Tree. 1997, 12, 474–478.

[41]Soloway, E. Learning to Program = Learning to Construct Mechanisms and Explanations [J]. Communications of the ACM. 2001, 29(9), 850-858.

Acknowledgement

Thanks to my Chinese tutor WANG Zhongjie. Thank you for your help and your advices during the period of writing the thesis. Thank you for patience.

Thanks to my supervisor Raphaël MARTAIN. Thank you for your patient guide always and thank you for your consideration during the period of my internship.

Thanks to my French tutor KunMean HOU. Thank you for your consideration during my study period in France.

Thanks to HIT and ISIMA. Because of the cooperation between them, I have the opportunity to become an exchange student and have the chance to study in France.

Thanks to everybody who helps me.

Resume

From 2005 to 2009, Major in software engineering in HIT (Harbin Institute of Technology) and obtain Bachelor's Degree of software engineering.

From 2009 to present, Exchange graduate student Harbin Institute of Technology of Software Engineering in China and ISIMA College of Engineering in Computer Science in France, double degree (Master's Degree in HIT and Engineering Degree in ISIMA).

Project experience:

From 2011 to present, system for analyzing the biomass and structure of pasture. Internship at INRA in Clermont-Ferrand (6 months).

From 2010 to 2011, Virtual Reality System. Project from APRV in Clermont-Ferrand (6 months). Programming environment: Unity. In this project, a virtual reality simulation system is build by Unity game engine. This system consists of the stereoscopic system and frustum systems. It simulates the users' views changing and directions changing to achieve the virtual reality effect.

From 2008 to 2009, OS of Music System in Car Navigator. Internship at NEUSOFT COMPANY (8 months). Programming language: C++. Develop the OS of a Music player in a car navigator for FUJI COMPANY (in Japan). It controls all the music applications in the navigator of the car. The details of this project are confidential.