

**MODELING PLANT COMPETITION WITH THE
GAPS OBJECT-ORIENTED DYNAMIC SIMULATION MODEL**

David G. Rossiter ^a

Susan J. Riha ^b

^a Soil Science Division, International Institute for Aerospace Surveys & Earth Sciences (ITC), Postbus 6, 7500 AA Enschede, the Netherlands. rossiter@itc.nl; <http://www.itc.nl/~rossiter>. **Corresponding author.**

^b Department of Soil, Crop, & Atmospheric Sciences, Emerson Hall, Cornell University, Ithaca, NY 14853 USA. sjr4@cornell.edu

Submitted to *Agronomy Journal* 22-December-1998

Revised version submitted 26-March-1999

Accepted 05-April-1999

PREPRINT April 1999

Copyright © 1999 by American Society of Agronomy, Inc.

MODELING PLANT COMPETITION WITH THE GAPS OBJECT-ORIENTED DYNAMIC SIMULATION MODEL

Abstract

Modeling inter-species competition is a natural application for dynamic simulation models of agricultural systems. We extended a dynamic simulation model of the soil-plant-atmosphere system, GAPS, to include the case where multiple plant species are growing in competition. The competition model was implemented as an object hierarchy which exchanges information with a crop model hierarchy by means of access procedures. Crop models, atmospheric procedures, and soil-plant water interface procedures required minimal change in order to support the competition model. We implemented two competition models: modified ALMANAC and 'winner-takes-all'. The modified ALMANAC model requires that crop models of competing plants simulate leaf area index, single-crop light interception, and canopy height. Using this information, the ALMANAC model partitions solar radiation among the competitors. GAPS is written in the object-oriented language Turbo Pascal.

Keywords

Object-oriented programming, dynamic simulation modeling, inter-species competition, GAPS model

Dynamic simulation models are well-established tools for understanding and managing agricultural systems (van Keulen & Wolf 1986; Driessen & Konijn 1992). Plant competition is an important part of agricultural systems, either by design (e.g. intercropping, relay cropping, agroforestry) or not (i.e. weeds). Inter-species competition is by nature dynamic (Graf *et al.*, 1990a; Graf *et al.*, 1990b; Weaver *et al.*, 1992), since the interactions between species and their effect on the total system can only be understood as they evolve over time in response to a forcing time series (weather), initial conditions, and the inertia of the system as represented by the soil and the plants themselves. Thus, modeling inter-species competition is a natural application for dynamic simulation models.

Many attempts have been made to model interspecies competition (Ball & Shaffer, 1993; Chikoye *et al.*, 1996; Graf *et al.*, 1990a; Graf *et al.*, 1990b; Kiniry *et al.*, 1992; Kropff & Spitters, 1992; Kropff *et al.*, 1992; McGowan *et al.*, 1996; Spitters & Aerts, 1983; Weaver *et al.*, 1994; Wiles *et al.*, 1996; Wilkerson *et al.*, 1990), and a few of these have been in the context of general-purpose dynamic simulation models originally developed to model the agricultural system, for example ALMANAC (Kiniry *et al.*, 1992) and SOYWEED (Wilkerson *et al.*, 1990). These two are examples of *monolithic* models: the processes and state of the competing species are an integral part of a single model, and can not be separated. We know of no modular competition models, other than that of McGowan *et al.* (1996), who, in their description of the APSIM modeling framework, mention in passing an 'arbiter' module that can partition resources among different crop models.

We extended a dynamic soil-plant-atmosphere simulation model, GAPS, to include the case where multiple plant species are growing in competition. The key innovations of this work are:

1. The competition model is implemented as an object hierarchy, which exchanges information with the existing crop model hierarchy by means of access procedures;
2. Competition for light is considered as a separate process, independent of any particular crop model;
3. Competition for water is an extension of existing water-uptake methods;

4. Run-time graphics provide a dynamic view of competition for light and water, and allow the program user to understand the results of different initial conditions and weather sequences.

We implemented two competition models for use in GAPS: a modified ALMANAC model (Kiniry *et al.*, 1992) and a 'winner-takes-all' model. This required substantial changes to details of the existing GAPS model, but did not require any changes to the model's modular and object-oriented structure.

This paper reports on the design objectives and structure of the GAPS simulation model and the adaptations necessary to model inter-species competition. Related papers report on the development and calibration of the competition model (McDonald & Riha, 1999a) and its application (McDonald & Riha, 1999b). We argue for a structured, modular, object-oriented design of complex simulation models, using as an example the work that was necessary to adapt GAPS to model plant competition. To this end, we have included some Turbo Pascal 5.5 (Borland International, 1990) code, simplified for didactic purposes. In the discussion of sample code, terms unique to object-oriented programming are *italicized*, and variable and method names are written in the `Courier` typeface. In the sample code itself, Pascal reserved words are written in **boldface**.

This paper is divided into three sections. First, we describe the GAPS computer program, emphasizing the structure of the simulation driver and the crop-model hierarchy. Second, we describe how GAPS was extended to simulate competition, respecting as far as possible the diverse crop models. Third, we describe the implementation of the modified ALMANAC competition model.

THE GAPS COMPUTER PROGRAM

'GAPS' is an acronym for 'General-purpose simulation model of the Atmosphere-Plant-Soil system'. GAPS is a program for microcomputers running the MS-DOS or Microsoft Windows 95 operating systems (Buttler & Riha, 1987; Riha *et al.*, 1994). It was originally

written by Buttlar in 1986 and has since been expanded and enhanced by a series of programmers. Its distinguishing features include the following:

1. GAPS represents soil, plant, atmospheric, and interface processes in a variety of ways. These representations can be selected independently of each other, so long as they have compatible conceptual models. Thus the model user can compare the effects of different ways of simulating the same phenomena. For example, potential evapotranspiration can be estimated by the Penman-Monteith, Priestly-Taylor, Linacre or Pan methods, and the user can decide which to use, independently of other modeling choices, as long as the data required by a method is available, and the method has been calibrated for a particular modeling situation.
2. It is menu-driven and includes an editor for preparing data and parameter files. There is also an MS-DOS command-line version with the same simulation procedures, which can be used for large numbers of repetitive simulations, and easily integrated with other computer programs for data entry, output, or analysis.
3. It can display a wide variety of graphs during the simulation run, as well as after the run is complete, allowing a dynamic understanding of the processes being simulated. In addition, values of state variables can be saved to a text file at any time step for later analysis or visualization.
4. It can simulate a sequence of crops and climates in a single- or multi-year simulation, and can simulate multiple crops growing at the same time (the principal subject of this paper).

GAPS was designed for use in research into environmental biophysics and for teaching the principles and practice of dynamic simulation modeling of the soil-plant-atmosphere biophysical system. Complete source code is distributed with GAPS, along with a comprehensive User's Manual (Riha *et al.*, 1994), so that the model user can see exactly how the program works. The program and manual are available at URL <http://wwwscas.cit.cornell.edu/sjr4/gaps.html> as well as by post from the second author.

Turbo Pascal (Borland International, 1990), now called Object Pascal and incorporated into the Delphi integrated development environment (INPRISE Corporation, 1998), is a non-standard dialect of the ISO Pascal programming language (ISO standard 7185:1990) (Jensen & Wirth, 1985) with which programs can be written to run under the MS-DOS and Microsoft Windows operating systems. It allows a large program to be broken up into modules (known as *units* in Object Pascal), within which there can be private variables and procedures that are not directly accessible by the rest of the program. In GAPS, the modules roughly correspond to components of the total system: atmosphere, soil, and plant, as well as administrative functions: simulation driver, global variables, file input and output, and graphic display.

GAPS has been used in a variety of research projects. These include simulating water fluxes in different agronomic systems (McIntyre *et al.*, 1996; McIntyre *et al.*, 1995; Buttler & Riha, 1992), developing an approach to estimating regional crop yields (Moen *et al.*, 1994), assessing the impact of temperature and precipitation variability on crop model predictions (Riha *et al.*, 1996), predicting the potential agronomic and economic impacts of gradual climate warming (Kaiser *et al.*, 1993), analyzing factors controlling water uptake at depth in semi-arid environments (McIntyre *et al.*, 1995), and comparing different representations of crop responses to elevated CO₂ (Melkonian *et al.*, 1998).

Structure of the simulation driver

GAPS is built around a simulation driver that controls how and when individual simulation modules are invoked. The simulation is time-driven, with procedures being called by the driver at various frequencies: yearly, daily, and sub-daily (e.g. hourly). The sub-daily frequency is the minimum time step for all processes and may be specified by the user. Figure 1 shows the structure of the simulation driver. Each of the procedures with names like ‘..._procedures’ are referred to as *subdrivers*, which contain a list of the simulation procedures to be carried out at that time resolution, depending on the model options selected

by the user. For example, the structure of the subdriver for the elementary time step is shown in Figure 2.

From this organization it follows that a particular simulation process, for example water flow in the soil, may be made up of several procedures, each corresponding to a time resolution. The several procedures for each module are linked through private *state variables* shared by the procedures of the module and invisible to procedures in other modules. In an example from the plant-available water uptake module (Figure 3), the total root length density (a private state variable) is calculated daily (because all the GAPS crop models only expand roots once a day), whereas water uptake is calculated every time step, using the total root length density to normalize per-layer demand. If information from one module is elsewhere in the program, it is obtained by an access function, as explained below).

Crop models in GAPS

Crop models in GAPS are arranged in an *object hierarchy* (Figure 4). The crop model *objects* contain data fields and the methods that manipulate them. Other sections of the program must access the fields via these methods. At the top level of the hierarchy is a generic crop model (`crop_model`) which defines the fundamental structure of all GAPS crop models, but which does not itself implement a complete crop model, and can not be selected by the GAPS user. Descended from this are four main crop models: (1) Stockle-Riha, adapted from the work of Stockle and Campbell (Stockle & Campbell, 1985; Stockle, 1985) by Imo Buttler and Susan Riha and extended to fast growing trees by Jennifer Phillips (Phillips, 1994); (2) SORKAM, adapted from the work of Rosenthal *et al.* (1989) by David Rossiter; (3) EPIC, adapted from the work of Williams *et al.* (1989) by Patrick Simoens; and (4) CONSTANT, written by Simoens to simulate a steady-state crop. Stockle-Riha is further divided into models for maize, wheat, and fast-growing trees. These models are quite different in many essential concepts. For example, in the Stockle-Riha models, photosynthesis is computed each time step for both the sunlit and diffuse-lit leaf areas,

whereas in EPIC, photosynthesis is computed daily, and no attempt is made to differentiate direct and diffuse sunlight. Thus, the Stockle-Riha models might be expected to more realistically simulate photosynthesis in situations where water becomes limiting during mid-afternoon hours, so that stomates close. In EPIC, this complication is not considered important. This diversity in conceptual level of the models has major implications for a common competition model, as will be discussed later.

Figure 5 shows some sample code for the generic crop model (`crop_model`), which is defined as an object in the fourth line of the code. Data are associated with this object, including state variables such as the leaf area index (variable `LAI`) and variable `myCropI` of type `Crops`, which the object uses at run time to determine its position in the array of active crops or weeds. The object `crop_model` has various *methods* (functions and procedures) associated with it that are listed as part of its type declaration. These methods implement the behavior (actions) of the crop model.

In the next fragment of sample code (Figure 6), another object, `SR_model`, is defined. This is an object of the type `crop_model`. Thus, it *inherits* all the data and methods of the `crop_model` object, for example `LAI` and `myCropI`. Additional data and procedures associated with `SR_model` are listed below it. A major advantage of the hierarchical organization is that so-called *virtual* methods defined at a higher level in the hierarchy need not (but may be) *over-ridden* by more specific methods. For example, the virtual procedure `End_growth` of the object `SR_model` over-rides the virtual procedure `End_growth` of the object `crop_model`. Thus an object of type `SR_model` will use its own `End_growth` procedure, instead of the one that it inherits from the generic crop model.

If a set of models share common methods, these methods need only be programmed once, and all descendent crop models will automatically inherit these methods. For example, all the Stockle-Riha models share the same `LimitRootingDepth` procedure, defined at the level of the object `SR_model`, but used by the three specific models one level down in the hierarchy.

GAPS takes advantage of Object Pascal's dynamic memory allocation features. Crop model objects are created and destroyed *dynamically*, that is, during the execution (as opposed to the compilation) of the program. They have no fixed memory location, but instead use memory that is allocated dynamically from the *heap*, which is memory that can be requested from the operating system as needed. Thus any number of crops can be specified at run time. Access to dynamic or virtual objects is by means of *pointer* variables, which contain a memory address of the object, not the object itself as with a conventional variable. Thus, in any code outside of the object which needs to access the object, the pointer must be *dereferenced*, i.e. the address is used to find the variable, by using the '^' Pascal operator to access the object itself. In the sample code (Figure 5 and Figure 6), this occurs immediately below the type declaration: the variables `crop_model_ptr` and `SR_model_ptr` refer to the memory addresses of their corresponding objects. These addresses are not established until run time.

In the last line of Figure 6, variable `TheCrop` is defined as an array of the type `Crop_Model_Ptr`. The length of the array depends on the number of crop objects (`Crops`, defined elsewhere as a program constant) that can be called during a simulation. The pointer variables in this array can be assigned to any object in the hierarchy; they are called *polymorphic* because their behavior varies with the actual object to which they are assigned. Procedure calls using these pointers will automatically call the virtual procedure that is defined the farthest down in the hierarchy. During program initialization, the pointers are assigned to crop models, according to the modeler's request.

An example of the use of polymorphic pointers and a virtual access function is shown in Figure 2. The function `Is_Transpiring` is used to determine whether a crop is transpiring, and the virtual procedure `TimeStep_Growth` is used to call the correct procedure in order to simulate crop growth for a single time step. For example, if the EPIC model had been selected to simulate the first crop in the array, the function `SE_model.is_transpiring` would automatically be called, because the pointer variable `TheCrop[1]` would be of type `SE_model_ptr`. The polymorphic pointer ensures that the correct version of the procedures

are called for each element of the array of active crops. Thus new crop models or variants may be implemented without any change to the simulation driver. The use of polymorphic pointers is also illustrated in Figure 3; here various state variables (e.g. `TotalRoots`, `FRoot`, `LRoot`) are referred to via such pointers.

SIMULATING COMPETITION IN GAPS

The major processes in inter-species competition are competition for light, water and nutrients. In GAPS, evapotranspiration, water uptake, and water movement in the soil are simulated every time step (user-selectable, from five minutes to six hours), whereas plant growth (changing LAI, canopy expansion, root expansion) is simulated only every day, so the competition model must operate at both daily and time-step scales. Photosynthesis is a special case: the Stockle-Riha models simulate this every time step, whereas EPIC and SORKAM simulate it only on a daily basis. GAPS does not simulate plant nutrition or soil chemistry, so it can not simulate competition for nutrients.

Design objectives

When adding new functionality to the existing program, we followed the following principles:

1. The same computer program should be used to simulate monoculture, competition, and fallow. Thus monoculture and fallow are just special cases of polyculture, and all the options of GAPS (e.g., the selection of evapotranspiration method, the decision on whether to model runoff or not) should be available to the modeler interested in simulating competition.
2. Crop models should be changed as little as possible. A crop model should not include any code that depends on whether the crop that it simulates is being grown as an intercrop or with a weedy competitor or not. It should contain no code that refers directly to a competitor's state, only to the availability of light and water as influenced by competition, if any. The crop models in GAPS originally assumed that the crop is being grown in monoculture, with exclusive access to available light and water. Therefore, these models

had to be changed somewhat. The challenge was to preserve their structure as much as possible, so that converting a crop model for use in multi-plant simulation is a simple and error-free exercise.

3. It should be possible to add different methods of simulating competition with minimum effort, and without disrupting the overall program structure.

Overview of the competition model

The competition module was implemented as a separate object hierarchy, which is called at appropriate times by the simulation driver to allocate limited resources to the various crops. To do this, competition objects need to determine the actual status of the crops, for which purpose they call methods defined in the crop model hierarchy. In return, the various crop objects can call the competition objects to determine the resource allocated to that crop. Competition between crops is automatically simulated if the user specifies more than one plant to be grown at a time.

All data structures having to do with crop files and crops as accessed by non-crop modules are one-dimensional arrays, indexed by an arbitrary crop sequence number. An input file specifies how competition should be modeled, including the cropping sequence, the years and dates of planting and harvest, and which competition model to use. Plant state variables (e.g., LAI, photosynthesis, transpiration) are implemented as data fields in the generic crop model object, so that each of the competing plants has its own state. Variables of the same name are also in the 'global' data structure, where they represent the aggregate of all crops, e.g., the total LAI and the total transpiration rate over all crops. The simulation driver calls the competition module at appropriate times, namely:

1. to initialize a competition model object at the beginning of simulation, before any crop model objects are defined;
2. to call the competition model at the beginning of each day, after daily atmospheric calculations but before any plant procedures;
3. to summarize plant state variables into 'global' data structures at the end of each day.

4. to dispose of the object at the end of simulation.

In principle, the competition model could also be called at each time step, but the actual competition models work on a daily frequency/

Crop models

Each crop model is responsible for maintaining certain information that is needed by the competition models. Figure 7 shows the variables which the crop model must make available to the competition model. We now discuss each of these.

Leaf Area Index

A crucial part of competition models is the simulation of leaf growth of the competing species. In GAPS, this task is complicated by the fact that different crop growth models have different ways of accounting for LAI. The Stockle-Riha and SORKAM models simulate LAI during growth as a function of accumulated top dry matter, which in turn is simulated as a function of photosynthesis, which in turn is simulated as a function of light intercepted by the canopy. (After a certain physiological stage, LAI declines as a function of days and a water-stress factor.) Therefore, the competition module has already affected LAI, once it has partitioned light according to transpiration demand and canopy interception.

However, the EPIC model simulates LAI during growth as a function of potential LAI (a parameter), a stress factor (computed dynamically from the moisture and temperature stresses), and the proportion of heat units accumulated to date, compared with those that must be accumulated to achieve full LAI (a parameter). Thus LAI is not related directly to biomass or photosynthesis in this model, and therefore the effect of competition must be simulated directly within the model. To do this, we adopted the approach taken in ALMANAC, an extension of EPIC (Kiniry *et al.*, 1992, equation 3): maximum potential LAI is decreased by a factor to account for competition. Here is one place where the design objective failed: we had to modify the structure of the crop model itself to account for competition. However, the change is backwards-compatible: if there is no competition (i.e.,

the plant population is optimal), the crop can achieve the same maximum LAI as before competition was added to the model.

The adjustment is done at crop model initialization and is based on the user-supplied plant density (plants ha⁻¹) and two calibration points on an S-shaped curve of the form:

$$S(x) = x / [x + \exp(y_1 - y_2 * x)]$$

The S-curve is used as an empirical representation of a response (here, maximum potential LAI) to a limiting variable (here, plant population), where the response follows the law of diminishing returns at both extremes of the limiting variable, and is most sensitive in the middle of the limiting variable's range.

GAPS calculates the curve parameters from two calibration points of the form (plant density ha⁻¹, proportion of maximum LAI), for example, (10 000, 0.2) and (40 000, 0.8).

These points should be established by experiment. If the calibration points are labeled (x₁, f₁) and (x₂, f₂), the analytical solution for the S-curve parameters is then:

$$y_2 = [\ln((-x_2 * (f_2-1)) / f_2) - \ln((-x_1 * (f_1-1))/f_1)] / [x_1 - x_2]$$

$$y_1 = \ln((-x_1 * (f_1-1)) / f_1) + y_2 * x_1$$

From the calibrated curve, GAPS computes the correction factor to reduce maximum potential LAI as a function of the simulated crop's plant density.

Canopy Height

Each crop model is responsible for simulating canopy height, which may be solicited by the competition module by calling the `GetHeight` method of the generic crop model. This virtual access procedure simply returns the current value of the `CanopyHeight` field of the generic crop model. However, it may be over-ridden by specific crop models. In particular, the EPIC model simulates canopy height as a function of the accumulated heat units as a proportion of heat units to maximum LAI. Also, the fast-growing tree model simulates canopy height as part of plant growth. Finally, the constant crop model has a constant height. For the remaining models (Stockle-Riha maize and wheat, SORKAM), we added an S-shaped growth function, which models height as a function of the proportion of heat units to a specific growth stage. The model user must supply two points on the curve, from which it may be

calibrated. This S-curve has the same structure as that as used in EPIC to correct maximum LAI for competition, as explained in the previous section.

A further correction is needed to ensure that plants do not have an unrealistically high height-to-weight ratio in the situation where biomass is reduced by competition. For models that explicitly model leaves and internodes, this would be accounted for already by the partitioning of photosynthate. However, the Stockle-Riha maize, Stockle-Riha wheat, modified SORKAM, and EPIC models in GAPS are not so detailed. For these, we limit the height achieved each day to a fraction of above-ground biomass, using an empirical factor with units $\text{m-height kg}^{-1}\text{-biomass}$, which is thus a required parameter for each crop model.

We did not model etiolation of shaded weeds caused by the near-infrared to red light ratio. In this situation, etiolated weeds may in fact emerge above the crop canopy, but will have very little leaf area on the elongated stem, and so will not significantly shade the crop.

Solar radiation

In all the crop models included in GAPS except the CONSTANT model, photosynthesis is computed as a function of photosynthetically active radiation (PAR), which is in turn a function of daily solar radiation (MJ m^{-2}). However, when several crops are grown together, they compete for solar radiation. Therefore, in place of a reference to the daily solar radiation `clim.SolRad[day]` as read from the climate file, the crop model calls a virtual access procedure `TheCompModel^.MySolRadA(MyCropI)` in the competition module. Variable `myCropI` is the crop object's own record of its sequence number in the list of active crops and `TheCompModel` is the pointer to the competition model. The competition module returns the solar radiation apportioned to the crop. At the beginning of each day, the simulation driver calls the competition model, which takes total solar radiation for the day, and partitions it into an array `SolRadA[]`, which is part of the competition model object's state. These values are returned to the crop models when the virtual access procedure is called. Then, the crop model determines how much of the incident radiation allocated to the crop is actually captured. Models may have different ways to do this. For example, the Stockle-Riha maize model explicitly estimates direct and diffuse radiation, whereas the EPIC model does not.

To partition solar radiation, a competition model needs some information from each crop model. For example, the ALMANAC model needs the daily LAI and canopy height of each active crop, as well as its potential-transpiration fraction (the proportion of solar radiation that it could use for transpiration). This flow of control and information is shown in Figure 8. The key point is that each module keeps track of its own state data, and supplies this data to other modules at their request.

The default is for no competition: a single crop gets all resources. In the default 'winner-takes-all' competition model the first-named crop model is allocated all the solar radiation. This allows the model user to see the effect of competition by changing a single modeling option, i.e., the choice of competition model.

Soil water-atmosphere interface procedures

The `PartitionETP` procedure (in the `atmoslib` unit) partitions the atmospheric demand (as a rate, kg water m⁻² land surface s⁻¹) into potential transpiration for all crops together, and potential soil evaporation. The partitioning depends on a composite potential-transpiration fraction, which is the fraction of light intercepted by the entire ground cover (all crops). This is supplied by the virtual access procedure of the competition model object, `GetTransFrac`. The potential soil evaporation is then used as the basis for calculating actual soil evaporation in the various soil water flow methods contained in the `soillib` unit.

Plant-soil water interface procedures

GAPS has three methods of simulating water uptake: two simple 'plant-available water' methods (`SimpleWaterUptake` and `EPICWaterUptake`), and a method where uptake is driven by water potentials from soil through root and stem to leaves and the atmosphere (`PDWaterUptake`). These methods are also contained in the `soillib` unit.

Competition for water is modeled within the uptake procedures, not as part of the competition model, because the modifications are natural extensions of the single-crop procedures. In the `SimpleWaterUptake` method, each soil layer's plant-available water (defined as water held between field capacity and permanent wilting point) is supplied to a

single growing plant in proportion to the plant's total root length density contained in the soil layer. In the `EPICWaterUptake` method, plant-available water is supplied in proportion to a decreasing exponential function with depth, with a user-defined decay parameter. In the `PDWaterUptake` method, each crop equilibrates its leaf and root water potentials with the soil water potential, and takes up water according to the potential gradient. The only change in these methods for a multi-crop simulation is that each plant's water uptake is simulated in turn, based on each crop's rooting density distribution or depth functions. There is no attempt to partition available water. In the plant-available water uptake methods, the amount of available water is reduced as it is taken up by each simulated crop in turn. In the potential-driven water uptake method, the soil water potential is not adjusted, but is assumed to be the same for all crops during the time step. These simplifications, which are similar to the original ALMANAC model, have minimal impact on the simulation, because they only affect water uptake during the single time step when the available water in a layer becomes depleted. The onset of water stress is delayed at most one time step.

THE MODIFIED ALMANAC MODEL

Kiniry *et al.* (1992) describe the ALMANAC competition model, which is based on the EPIC crop models (Williams *et al.*, 1984; Williams *et al.*, 1989) and the competition model described by Spitters & Aerts (1983). We used the ideas behind ALMANAC, with some modifications, as the basis of a competition model. First, the model was made more general, to allow any number of competing crops, instead of the two (crop and weed) allowed by ALMANAC. Second, we used Wallace's (1995) method for light partitioning in preference to the Spitters & Aerts method used in ALMANAC. Third, some aspects of ALMANAC were implemented in the EPIC crop model itself and in the three water uptake procedures, rather than in the competition module; these have been explained above. We call what is left in the competition module the 'modified ALMANAC' model in the GAPS hierarchy of competition

models. We now describe in detail how we modeled light interception by each crop and total canopy light interception in this model.

These are implemented as part of the `BeginDay` procedure, which is called by the simulation driver at the beginning of each day when there is at least one active crop model. This is done after solar angles are computed but before any crop events, including planting and harvesting.

First, the potential-transpiration fraction for each crop (the proportion of light that is intercepted by the crop in monoculture) must be supplied by an access procedure in each crop model. Its complement, which is the fraction of light not intercepted, is recorded in array `TF`:

```
if (TheCrop[crop_i] <> nil) then
    TF[crop_i] := 1 - TheCrop[crop_i]^GetTransFrac;
```

where `TheCrop[crop_i]` is a pointer to crop model with sequence number `crop_i`. The `GetTransFrac` access function is defined as a virtual object method in the generic crop model, which implements it as a simple Beer's Law function of LAI:

```
GetTransFrac := 1 - exp(k * LAI)
```

where k is a (constant) extinction coefficient. Since all crop models are defined as descendents of the generic crop model, any crop model that can accept this function and which models LAI need not over-ride the virtual method; this illustrates the power and economy of an object-oriented model.

The model keeps a running sum of exponentials (i.e., a product of fractions) for total transmission:

```
TTF := TTF * TF[crop_i]
```

where `TTF` was initialized to unity, and `TF` is the single-crop transmission fraction. The model then queries each crop model for its current LAI:

```
LA[crop_i] := TheCrop[crop_i]^LAI
```

A crop model may use its own method to simulate light interception. In fact, the Stockle-Riha model uses a polynomial function of LAI, and SORKAM uses a square-root function of LAI. However, since the competition model assumes that each crop model

expresses light interception by Beer's law, for any given LAI, the same light interception could have been obtained from the Beer's Law equation with an appropriate k . So, to express each crop's interception in the same terms, the presumed k is calculated with the inverse of Beer's law:

$$K[\text{crop}_i] := -\ln(\text{TF}[\text{crop}_i]) / \text{LA}[\text{crop}_i]$$

The model queries each crop model for its current canopy height, using an access procedure, and keeps a running sum of the heights:

$$\begin{aligned} H[\text{crop}_i] &:= \text{TheCrop}[\text{crop}_i]^\wedge.\text{GetHeight}; \\ \text{sumH} &:= \text{sumH} + H[\text{crop}_i] \end{aligned}$$

To partition the solar radiation to a crop using the method of Wallace (1995), the model first determines f_s , the light interception fraction when the crop is completely shaded by the other crops, with Wallace's equation (6):

$$f_s := (\text{TTF} / \text{TF}[\text{crop}_i]) * (1 - \text{TF}[\text{crop}_i])$$

Since TTF is the product of the interception fractions, the ratio $\text{TTF} / \text{TF}[\text{crop}_i]$ is the product of the interception fractions of all crops except crop crop_i . This is then multiplied by the potential-transpiration fraction of the crop in question, i.e. the light intercepted by that crop. Thus f_s is the fraction of light that would be incident on a completely-shaded crop.

The other extreme is the case where the crop in question shades all other crops, in which case it would receive all the light it could intercept, i.e. its own potential-transpiration fraction, $1 - \text{TF}[\text{crop}_i]$. Wallace proposes linear interpolation between these two extremes, using the height ratio (his equation (10)):

$$f := H[\text{crop}_i] / \text{sumH}$$

This assumes that plant leaf density is uniform over the height of the plant, which is a reasonable simplification for many crop and weed species. Then the fraction of total radiation intercepted by the crop is found by interpolation:

$$\text{SolRadFrac}[\text{crop}_i] := (f_s + (f * ((1 - \text{TF}[\text{crop}_i]) - f_s)))$$

and the actual radiation intercepted, in MJ m^{-2} , is found by multiplying this fraction by the day's measured solar radiation:

```
SolRadA[crop_i] := SolRadFrac[crop_i] * clim.SolRad[real_day]
```

The two arrays `SolRadFrac` and `SolRadA` are stored in the competition object, and are provided to the various crop models via the virtual access method `MySolRadA`, which is defined in the base competition model.

Following Wallace (1995), the total light interception of the canopy for all crops taken together is modeled as the exponential sum of the individual Beer's Law interceptions. This is reported by the competition model's virtual access function `GetTransFrac`, which is simply the complement of the product of the transmissions of the several crops:

```
TransFrac := 1 - TTF
```

As Wallace points out, this implies that "...total light intercepted by the mixture [of species] is independent of the individual species heights, whereas the individual fractions... are not."

VISUALIZING COMPETITION

Both the DOS and Microsoft Windows 95 versions of GAPS include a wide range of user-selectable run-time graphs, which allow modelers to visualize the simulation. For visualizing competition, several graphs are particularly relevant: LAI, canopy height, rooting depth, potential and actual transpiration, stress index, per-day and cumulative growing-degree days, top and root dry matter, and grain yield and moisture. All of these can show all crops on the same graph. A particularly informative graph shows daily theoretical and measured solar radiation, and the solar radiation intercepted by each crop, as a stacked bar graph. The graphing routines call virtual access functions in the crop objects to determine the values of variables to be graphed.

An example is shown in Figure 9. A maize crop is modeled with the Stockle-Riha model, and a generic C4 weed with the EPIC model. The graphs show how radiation is partitioned, and how top dry matter and LAI develop over time.

CONCLUSION

It proved possible to modify the dynamic simulation modeling program GAPS to model plant competition, without serious disruption to the existing program structure. This was due to the program's modular structure (e.g., separation of atmospheric, soil, plant and interface

procedures; 'need to know' placement of state variables) as well as to the organization of diverse crop models in an object hierarchy. The competition module organizes competition models as an object hierarchy, making it possible to add specializations of the competition model that was implemented (modified ALMANAC) or to add new competition models, with minimal further change in the crop models.

REFERENCES

- Ball, D.A. & Shaffer, M.J. (1993). Simulating resource competition in multispecies agricultural plant communities. *Weed Research* **33**, 299-310.
- Borland International (1990). Computer program: *Turbo Pascal 5.5*. Scotts Valley, CA, Borland International.
- Buttler, I. & Riha, S.J. (1987). General purpose simulation model of water flow in the soil-plant-atmosphere continuum. *Applied Agricultural Research* **2**, 230-234.
- Buttler, I.W. & Riha, S.J. (1992). Water fluxes in Oxisols: A comparison of approaches. *Water Resources Research* **28**, 221-229.
- Chikoye, D., Hunt, L.A. & Swanton, C.J. (1996). Simulation of competition for photosynthetically active radiation between common ragweed (*Ambrosia artemisiifolia*) and dry bean (*Phaseolus vulgaris*). *Weed Science* **44**, 545-554.
- Driessen, P. M. and N. T. Konijn (1992). *Land-use systems analysis*. Wageningen, Wageningen Agricultural University, Department of Soil Science & Geology.
- Graf, B., Gutierrez, A.P., Rakotobe, O., Zahner, P. & Delucchi, V. (1990a). A simulation model for the dynamics of rice growth and development: Part II - The competition with weeds for nitrogen and light. *Agricultural Systems* **32**, 367-392.
- Graf, B., Rakotobe, O., Zahner, P., Delucchi, V. & Gutierrez, A.P. (1990b). A simulation model for the dynamics of rice growth and development: Part I - The carbon balance. *Agricultural Systems* **32**, 367-392.
- INPRISE Corporation (1998). Computer program: *Delphi 4*. Scotts Valley, CA, INPRISE Corporation.
- Jensen, K. & Wirth, N. (1985). *Pascal user manual and report, revised for the ISO Pascal standard*, 3rd ed. Springer-Verlag, New York.
- Kaiser, H.M., Riha, S.J., Wilks, D.S., Rossiter, D.G. & Sampath, R. (1993). A farm-level analysis of economic and agronomic impacts of gradual climate warming. *American Journal of Agricultural Economics* **75**, 387-398.
- van Keulen, H. and J. Wolf, Ed. (1986). *Modelling of agricultural production: weather, soils and crops*. Simulation Monographs. Wageningen, PUDOC.
- Kiniry, J.R., Williams, J.R., Gassman, P.W. & Debaeke, P. (1992). A general, process-oriented model for two competing plant species. *Transactions of the ASAE* **35**, 801-810.
- Kropff, M.J. & Spitters, C.J.T. (1992). An eco-physiological model for interspecific competition, applied to the influence of *Chenopodium album* L. on sugar beet. I. Model description and parameterization. *Weed Research* **32**, 437-450.
- Kropff, M.J., Spitters, C.J.T., Schnieders, B.J., Joenje, W. & de Groot, W. (1992). An eco-physiological model for interspecific competition, applied to the influence of *Chenopodium album* L. on sugar beet. II. Model evaluation. *Weed Research* **32**, 451-463.

- McDonald, A. & Riha, S.J. (1999a). Model of crop-weed competition applied to maize – *Abutilon theophrasti* system. I. Model description. Submitted to *Weed Research*.
- McDonald, A. & Riha, S.J. (1999b). Model of crop-weed competition applied to maize – *Abutilon theophrasti* system. II. Assessing the impact of climate: implications for economic thresholds. Submitted to *Weed Research*.
- McGowan, R.L., Hammer, G.L., Hargreaves, J.N.G., Holzworth, D.P. & Freebairn, D.M. (1996). APSIM: a novel software system for model development, model testing & simulation in agricultural systems research. *Agricultural Systems* **50**, 255-271.
- McIntyre, B.D., Riha, S.J. & Flower, D.J. (1995). Water uptake by pearl millet in a semiarid environment. *Field Crops Research* **43**, 67-76.
- McIntyre, B.D., Riha, S.J. & Ong, C.K. (1996). Light interception and evapotranspiration in hedgerow agroforestry systems. *Agricultural and Forest Meteorology* **81**, 31-40.
- Melkonian, J., Riha, S.J. & Wilks, D.S. (1998). Simulation of elevated CO₂ effects on daily net canopy carbon assimilation and crop yield. *Agricultural Systems* **58**, 87-106.
- Moen, T.N., Kaiser, H.M. & Riha, S.J. (1994). Regional yield estimation using a crop simulation model: Concepts, methods, and validation. *Agricultural Systems* **46**, 79-92.
- Phillips, J. (1994). *Water Use in Eucalyptus*. PhD Thesis, Cornell University, Ithaca, NY. 156 pp.
- Riha, S.J., Rossiter, D.G. & Simoens, P. (1994). *GAPS: General-purpose Atmosphere-Plant-Soil Simulator. Version 3.0 User's Manual*. SCAS Teaching Series, Cornell University, Department of Soil, Crop & Atmospheric Sciences, Ithaca, NY.
- Riha, S.J., Wilks, D.S. & Simoens, P. (1996). Impact of temperature and precipitation variability on crop model predictions. *Climatic Change* **32**, 293-311.
- Rosenthal, W.D., Vanderlip, R.L., Jackson, B.S. & Arkin, G.F. (1989). *SORKAM: a grain sorghum crop growth model*. Texas Agricultural Experiment Station. TAES Computer Software Documentation Series MP1669
- Spitters, C.J.T. & Aerts, R. (1983). Simulation of competition for light and water in crop-weed associations. *Aspects of Applied Biology* **4**, 467-483.
- Stockle, C.O. (1985). *Simulation of the effect of water and nitrogen stress on growth and yield of spring wheat*. Ph.D. Thesis, Washington State University, Pullman, WA.
- Stockle, C.O. & Campbell, G.S. (1985). A simulation model for predicting effect of water stress on yield: an example using corn. *Advances in Irrigation* **3**, 283-323.
- Wallace, J.S. (1995). Towards a coupled light partitioning and transpiration model for use in intercrops and agroforestry. In: *Ecophysiology of tropical intercropping* (ed. H. Sinoquet & P. Cruz). INRA, Paris, pp. 153-162.
- Weaver, S.E., Kropff, M.J. & Cousens, R. (1994). A simulation model of competition between winter wheat and *Avena fatua* for light. *Annals of applied biology* **124**, 315-331.
- Weaver, S.E., Kropff, M.J. & Groeneveld, R.M.W. (1992). Use of ecophysiological models for crop-weed interference: the critical period of weed interference. *Weed Science* **40**, 302-307.

Wiles, L.J., King, R.P., Schweizer, E.E., Lybecker, D.W. & Swinton, S.M. (1996). GWM: General Weed Management model. *Agricultural Systems* **50**, 355-376.

Wilkerson, G.G., Jones, J.W., Coble, H.D. & Gunsolus, J.L. (1990). SOYWEED: a simulation model of soybean and common cocklebur growth and competition. *Agronomy Journal* **82**, 1003-1010.

Williams, J.R., Jones, C.A. & Dyke, P.T. (1984). A modeling approach to determining the relationship between erosion and soil productivity. *Transactions of the ASAE* **27**, 129-144.

Williams, J.R., Jones, C.A., Kiniry, J.R. & Spanel, D.A. (1989). The EPIC crop growth model. *Transactions of the ASAE* **32**, 497-511.

FIGURES

FIGURE 1
Main simulation driver

```
begin
  Begin_simulation_procedures;
  for year := first_year to last_year do begin
    Begin_year_procedures;
    for day := first_day to last_day do begin
      Begin_day_procedures;
      for time_step := 1 to num_of_time_steps do
        TimeStep_procedures;
      end;
      End_day_procedures;
    end; { for each day }
    End_year_procedures;
  end; { for each year }
  End_simulation_procedures;
end;
```

FIGURE 2
Simulation sub-driver for each time step

```

procedure TimeStep_procedures;
  begin
    { atmospheric state }
      SolarAngle; Air_Temperature; Distribute_Precipitation;
    { soil temperature }
      case SoilTemp_method of
        Variable: Numeric_Soil_Temperature_TimeStep;
        Harmonic: Harmonic_Soil_Temperature_TimeStep;
      end;
    { potential ETP }
      case ETP_method of
        Priestley      :   Priestley_Taylor_ETP_TimeStep;
        Penman        :   Penman_ETP_TimeStep;
        Linacre       :   Linacre_ETP_TimeStep;
        Pan           :   Pan_ETP_TimeStep;
      end;
    PartitionETP;
    if ActiveCrops > 0 then begin { at least one crop is active }
      { competition model }
        TheCompModel^.TimeStep;
      { grow the plants }
        for crop_i := 1 to global.MaxCrops do
          if (TheCrop[crop_i] <> nil)
            and (TheCrop[crop_i]^is_transpiring) then
              TheCrop[crop_i]^TimeStep_Growth;
          { take up water }
            case Uptake_method of
              Potential_driven : PDWaterUptake_TimeStep;
              Plant_available  : SimpleWaterUptake_TimeStep;
              Epic_water_uptake : EPICWaterUptake_TimeStep;
            end; { Uptake_method }
          end; { at least one crop is active }
      { soil water flow }
        case Flow_method of
          Richards_Equation : Richards_Equation_TimeStep;
          Tipping_Bucket   : Tipping_Bucket_TimeStep;
          ...
        end; { Flow_method }
      { summarize water relations for the time step }
        WaterBudget;
    end; { TimeStep_Procedures }
  
```

FIGURE 3
Simple water uptake, implemented at two time resolutions

```

unit soilib;
...
{ module state variables, visible only to all procedures in the module }
var
  TotalRoots : single; { total roots, all crops }
...
procedure SimpleWaterUptake_Daily; { called at the beginning of every day }
var
  crop_i : crops;
  layer : layers;
  sum : double; { roots of one crop }
begin
  { update the total root content once a day }
  TotalRoots := 0;
  for crop_i := 1 to MaxCrops do
    if (TheCrop[crop_i] <> nil) then begin
      sum := 0;
      for layer := TheCrop[crop_i]^crop.FRoot to TheCrop[crop_i]^LRoot do
        sum := sum + TheCrop[crop_i]^crop.RootDens[layer]*LayThick[layer];
      TheCrop[crop_i]^TotalRoots := sum;
      TotalRoots := TotalRoots + sum
    end { if there is a crop }
  ...
end; { SimpleWaterUptake_Daily }

procedure SimpleWaterUptake_TimeStep; { called at each time step }
var
  crop_i : crops; layer : layers;
  r_frac: double; { fraction of total roots in a layer }
begin
  for layer := MinLayer to sim.LRoot do begin
    LayerAvail[layer] := KgFines[layer] * (sim.WN[layer] - soil.DLL[layer]);
  end; { for each layer }
  for crop_i := 1 to global.MaxCrops do
    if TheCrop[crop_i] <> nil and TheCrop[crop_i]^PotTrans > 0 then
      for layer := TheCrop[crop_i]^crop.FRoot to TheCrop[crop_i]^LRoot do
        { apportion total demand by the plant according to its proportion of
          total roots in this layer }
        r_frac := (LayThick[layer]*TheCrop[crop_i]^crop.RootDens[layer])
          / TheCrop[crop_i]^TotalRoots;
  ...
end; { SimpleWaterUptake_TimeStep }

```

FIGURE 4
The crop model hierarchy

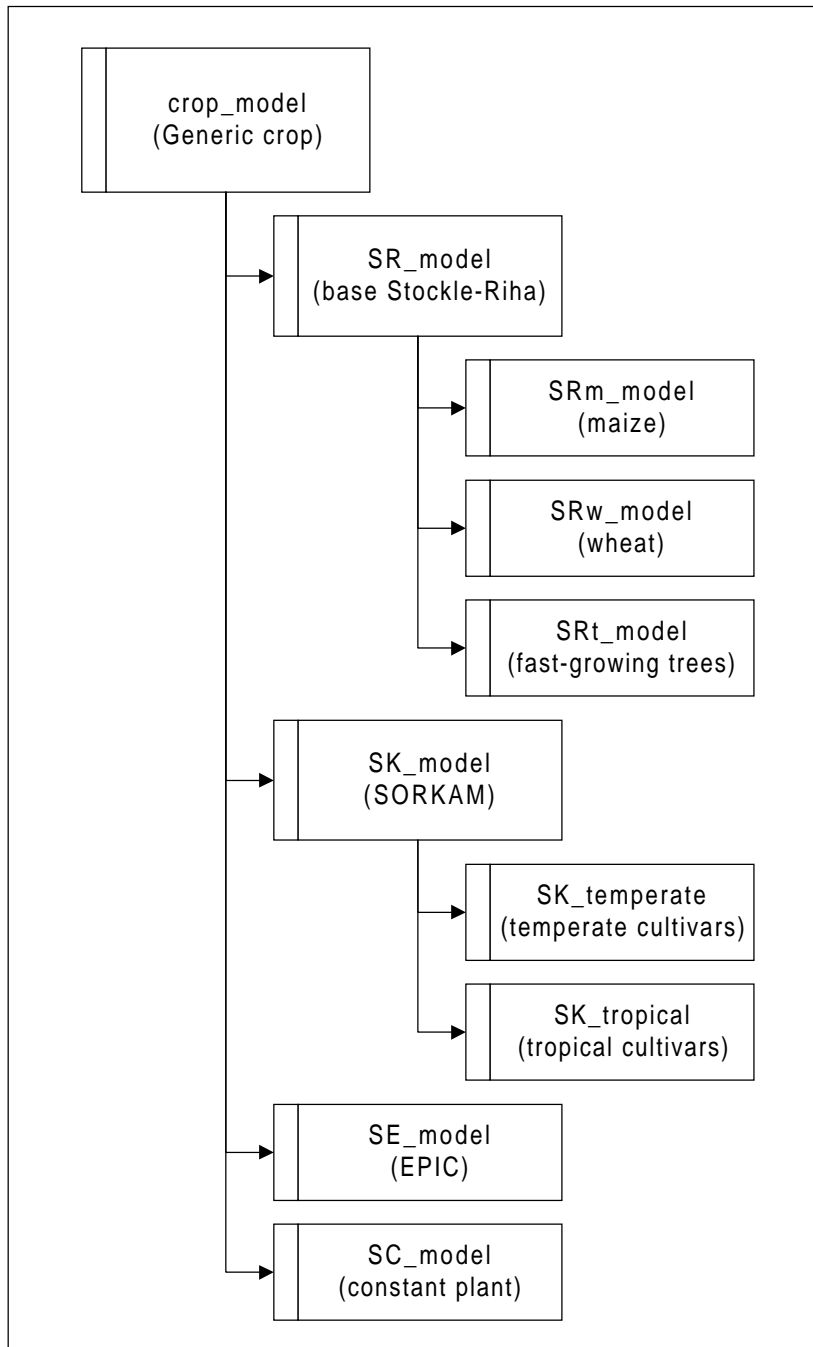


FIGURE 5
Part of an object hierarchy: the generic crop model

```

type
crop_model_ptr = ^crop_model; { general crop model }
crop_model = object
  myCropI : Crops; { this crop's position in the array of active crops }
  { generic state variables }
  LAI : single;          { Leaf Area Index }
  RootingDepth : single; { cm }
  PotTrans, ActTrans : single; { transpiration rates, kg m-2 s-1 }

  { procedures to create & destroy an object of this type }
  constructor Init(crop_i : Crops);
  destructor Done; virtual;

  { procedures called from any module }
  function GetTransFrac : single; virtual;
  function GetHeight : single; virtual;

  { procedures called by the simulation driver }
  procedure Begin_growth; virtual; { - one time, at the beginning of growth }
  procedure End_growth; virtual; { - one time, at end of growth }
  { - at the beginning of each day between sowing and maturity }
  procedure Daily_Growth_Begin; virtual;
  { - at the end of each day between sowing and maturity }
  { - after all daily atmospheric and soil procedures & the day's time-steps }
  procedure Daily_Growth_End; virtual;
  { - for each time step of each day between emergence and maturity }
  { - after hourly atmospheric procedures, before water flow }
  procedure TimeStep_Growth; virtual;

  { private procedures called within the object }
  { - effects of elevated CO2 on daily dry matter production }
  procedure CO2Response_Function; virtual;
end; { type crop_model }

```

FIGURE 6
Part of an object hierarchy: Stockle-Riha crop model

```

type
  SR_model_ptr = ^SR_model; { Stockle-Riha model - generic }
  SR_model = object(crop_model)
  { -- model constants }
  TempCoeff : single; { convert PAR to PS: linear factor}
  ...
  { -- entry points from simulation driver }
  procedure End_growth; virtual;
  procedure Daily_Growth_Begin; virtual;
  procedure Daily_Growth_End; virtual;
  procedure TimeStep_Growth; virtual;

  { -- private procedures called within the object }
  { simulated daily }
  procedure LimitRootingDepth;
    { simulated every time step }
  procedure Light_Interception; virtual;
  procedure Max_PhotoSynthesis; virtual;
  procedure Limit_PhotoSynthesis; virtual;
  { -- access procedures }
  function PSTempFac(AirTemp : single) : single; virtual;
  function GetHeight : single; virtual;
end; { SR_model }

type
  SRm_model_ptr = ^SRm_model; { Stockle-Riha maize model }
  SRm_model = object(SR_model)
  { heat units to each growth stage }
  AccDD_to : array[SRm_GrowthStages] of integer;
  ...
  { -- entry points from simulation driver }
  procedure End_growth; virtual;
  procedure Daily_Growth_End; virtual;

  { simulated daily }
  procedure Dry_Matter_Accumulation;
  procedure Yield; { optional }
  { simulated every time step }
  procedure Max_Photosynthesis; virtual;
  procedure Limit_PhotoSynthesis; virtual;

  { access procedures }
  function PSTempFac(AirTemp : single) : single; virtual;
  function GetTransFrac : single; virtual;
  function GetHeight : single; virtual;
  ...
  { -- private procedures called within the object }
  function Growth_Stages : SRm_GrowthStages;
end; { SRm_model }

{ crop model handles }
var
  TheCrop : array[Crops] of Crop_Model_Ptr; { nil if no crop object }

```

FIGURE 7

State variables from crop models used by the competition module

Variable	Meaning	Frequency	How accessed
LAI	Current Leaf Area Index	daily	a field in the generic crop object
TransFrac	Proportion of light intercepted by the canopy, single-crop	daily, depends on LAI	an object method GetTransFrac, which is assumed to have the form $1 - \exp(-k * LAI)$
CropHeight	Current height	daily	an object method GetHeight

FIGURE 8
Partitioning solar radiation: information flow

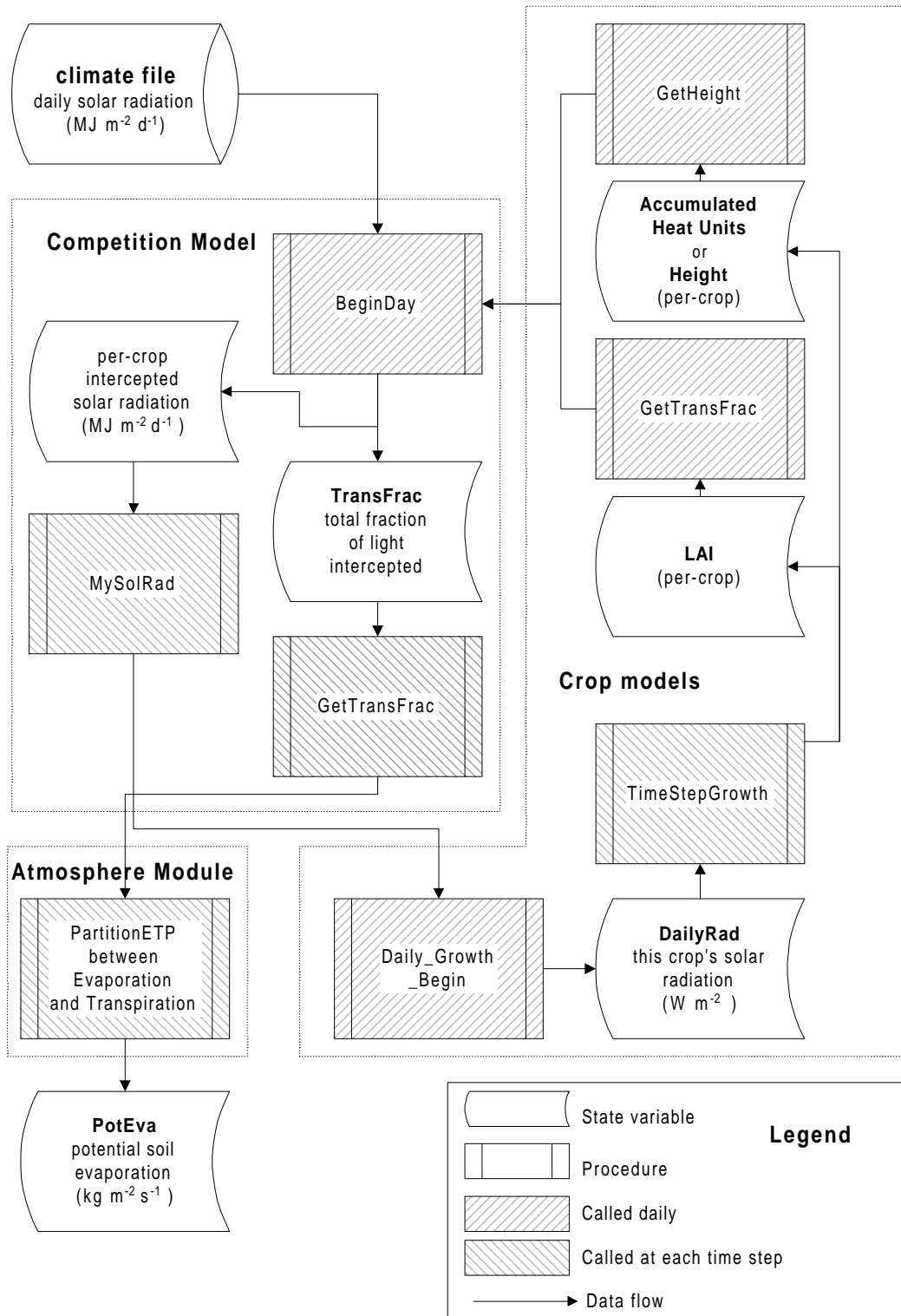


FIGURE 9
Run-time graphics

