Module 4

SELES – Driving Models

Scenario Scripts

Andrew Fall

Landscape Systems Ecologist

Gowlland Technologies Ltd.

April 2024

Module 4 Objectives

What you can expect to learn from this module:

- SELES
  - How to open and run an existing SELES model
  - How to change parameters and other inputs to an existing model
  - How to control simulations
  - How to modify inputs to an existing model to apply it in a different landscape

➢ See SELES User Documentation: Part 3 - sections 1, 2 and 3

Running Existing Models

overview

- Running existing models and scenarios
  - manually via the user interface (common during testing)
  - automatically via scenario scripts (common for experiments)

- Managing scenario scripts (SELES scenario language)
  - Changing parameter values, input layers, input tables, etc.
  - Controlling output locations
  - Designing experiments
  - Adapting models to new study areas

Running Existing Models

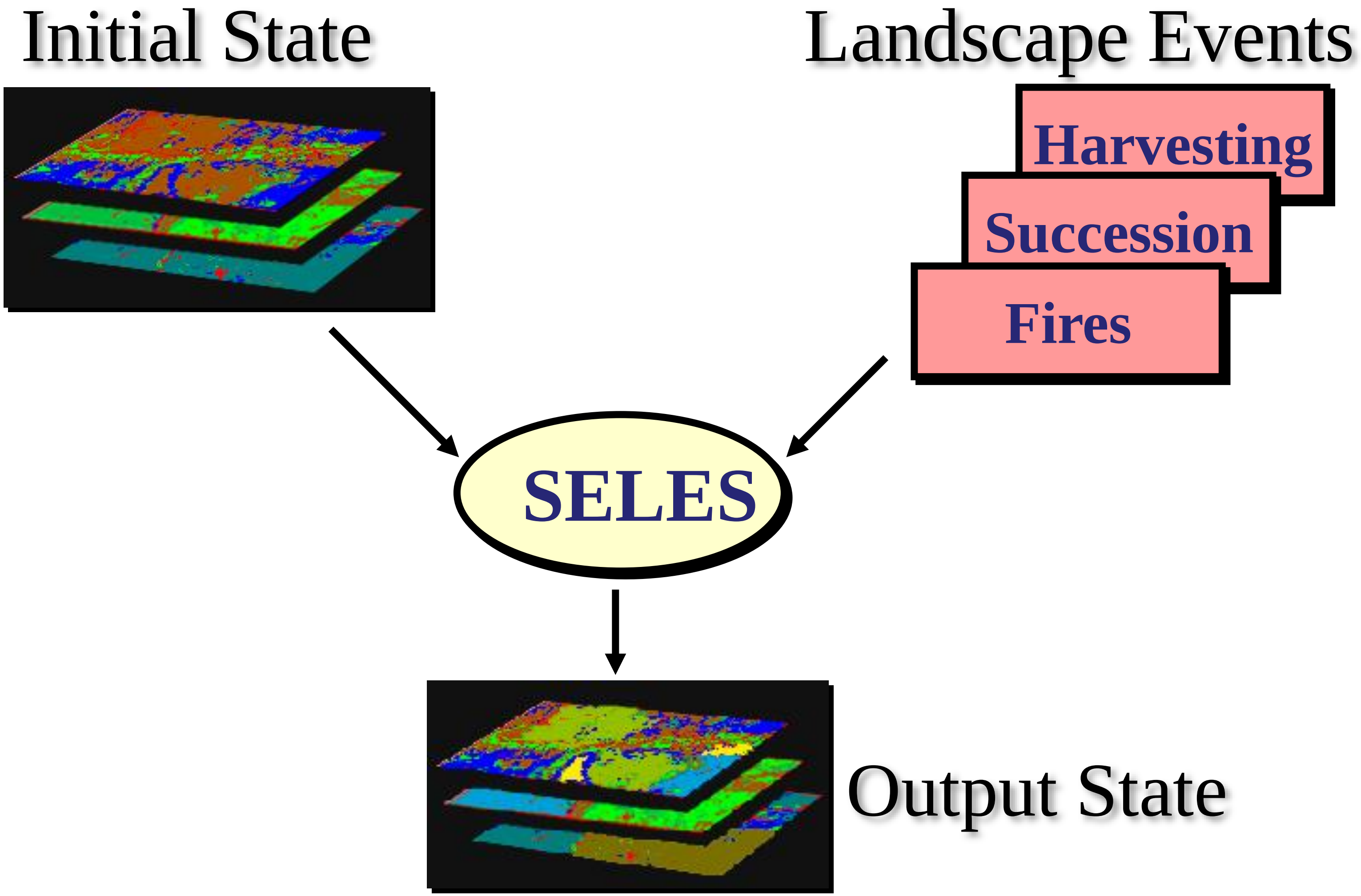
basic steps

- Open existing scenario script file (.scn files) in SELES

- IF simulation is started manually:
  - scenario will
    - load all necessary layers and models
    - set default parameter values
    - move to defined output folder
    - modify display state of views
  - need to run model via simulation dialog

- IF simulation is started automatically
  - scenario will additionally start simulation

SELES Scenario Structure



Initial State

Landscape Events

Harvesting

Succession

Fires

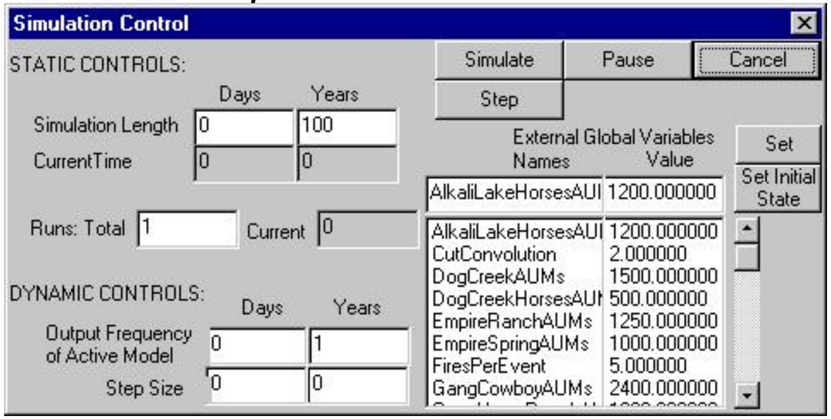
SELES

Output State

Running Existing Models

manual simulation control

- Set simulation duration
- Control buttons
  - *Simulate/Stop* toggle:
    - *Simulate* (simulation not running): start simulation
    - *Stop (*simulation running): early termination
  - *Step (*simulation not running or simulation paused):
    - start/continue simulation and pause after *Step size* time units
  - *Pause/Continue* toggle:
    - *Pause* (simulation running): temporarily halt simulation
    - *Continue* (simulation paused): continue simulation until end



Simulation Control

STATIC CONTROLS:

	Days	Years
Simulation Length	0	100
CurrentTime	0	0

Runs: Total 1 Current 0

DYNAMIC CONTROLS:

	Days	Years
Output Frequency of Active Model	0	1
Step Size	0	0

External Global Variables

Names	Value
AlkaliLakeHorsesAUI	1200.000000

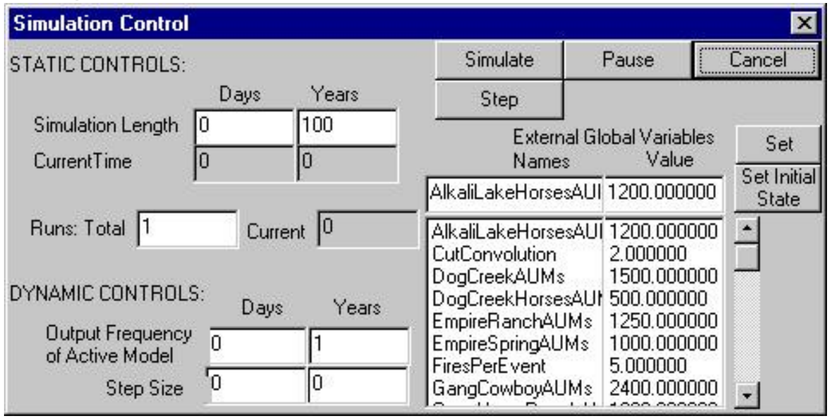
AlkaliLakeHorsesAUI	1200.000000
CutConvolution	2.000000
DogCreekAUMs	1500.000000
DogCreekHorsesAUI	500.000000
EmpireRanchAUMs	1250.000000
EmpireSpringAUMs	1000.000000
FiresPerEvent	5.000000
GangCowboyAUMs	2400.000000

Running Existing Models

manual simulation control

- Changing parameters
  - select variable in list and modify value
  - Press:
    - *Set* to change *current* value (useful while a simulation is running)
      - » won’t affect initial value loaded at simulation startup
      - » ∴ won’t have any effect if simulation is not running
    - *Set Initial state* to change *initial* value (useful when simulation not running)
      - » won’t have an affect until next simulation is started
      - » ∴ won’t affect a currently running simulation

- *Output frequency:*
  - Changes refresh rate of current view



STATIC CONTROLS:

	Days	Years
Simulation Length	0	100
CurrentTime	0	0

Runs: Total 1 Current 0

DYNAMIC CONTROLS:

	Days	Years
Output Frequency of Active Model	0	1
Step Size	0	0

External Global Variables

Names	Value
AlkaliLakeHorsesAUI	1200.000000

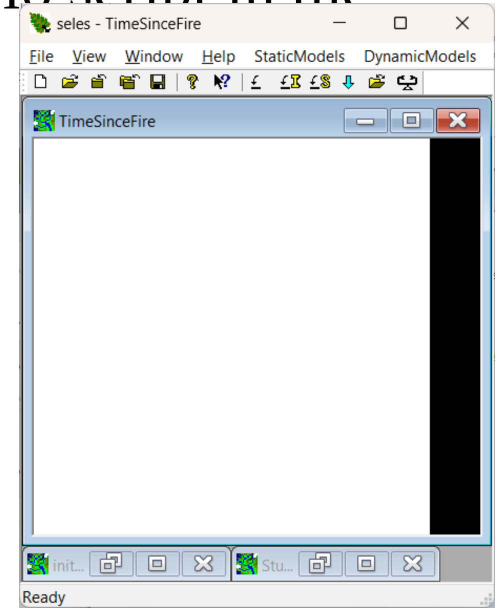
Names	Value
AlkaliLakeHorsesAUI	1200.000000
CutConvolution	2.000000
DogCreekAUMs	1500.000000
DogCreekHorsesAUI	500.000000
EmpireRanchAUMs	1250.000000
EmpireSpringAUMs	1000.000000
FiresPerEvent	5.000000
GangCowboyAUMs	2400.000000

Hands-on

loading a simple fire model

First steps

- Download and install SELES
- Download and unzip the tutorial model files
  - The main model files for this module are in the “SimpleFireModel” folder
- Start SELES and open FireTopDown.scn scenario script in the Scenarios folder
- Should look something like this:



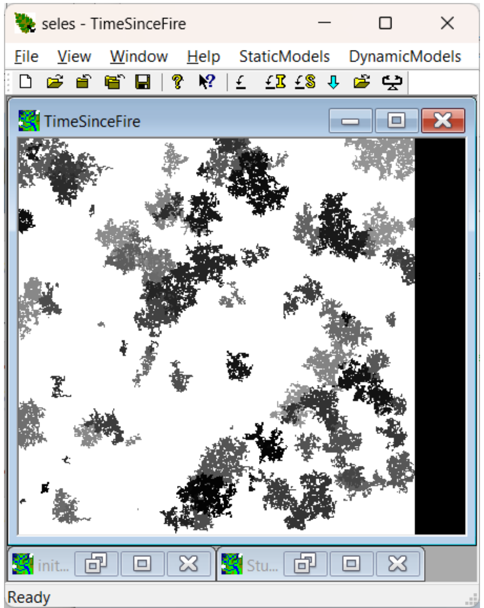
➢ This shows the start state (a simulation hasn’t yet been run)

Hands-on

running a simple fire model

Open the Simulation dialog (DynamicModels menu: Simulate or the blue down arrow on the toolbar)

- The default Simulation Length is 10 kilosteps (10,000 steps)
- The global variables list shows variables defined in the model (these may be input parameters, tracking variables and/or outputs)
- Press Simulate to start the model running
- After some steps, should look something like this:
- Note 1: load the legend (View menu: Show Legend) to see that black means recently burned which lightens as cells age
- Note 2: Set "Slowdown" to 10 to slow it down a bit



Hands-on

description of the "simple top-down fire model"

The model state-space includes:

- A static StudyArea layer (to define the area of interest);
- A dynamic TimeSinceFire layer (approximately age); and
- Two global variable parameters: MeanFiresPerYear and MeanFireSize

There are two modelled processes:

(i) Aging: each step, increase each cell age by 1 (TimeSinceFire = TimeSinceFire + 1)

(ii) Fire: each step:

- ○ Ignition: randomly select the number of fires ($\geq$ 0) from an exponential distribution (mean MeanFiresPerYear), at randomly selected locations;
- ○ Target size: For each fire, randomly select an *extent* from an exponential distribution (mean MeanFiresSize);
- ○ Spread: iteratively spread to a random number of the 4 cardinal neighbours (but not burnt this step) until the target size is reached
- ○ Effect (on burning): set TimeSinceFire = 0 and ssum the area burned (AreaBurned)

Hands-on

modifying parameters

Two main parameters are: MeanFiresPerYear and MeanFireSize

- Click on one of these and change its value in the field at the top of the list
  - Press Set if the simulation is running (this will change the current value and so affect the simulation; reset to initial value when a new simulation starts)
  - Press Set Initial State if the simulation is not running (this will change the value used to initialize the variable at simulation start up, but won’t affect a currently running simulation)

➢ See how the TimeSinceFire layer changes with fewer/more or smaller/larger fires. Also notice how the FireCycle variable changes.

Running Existing Models

understanding models

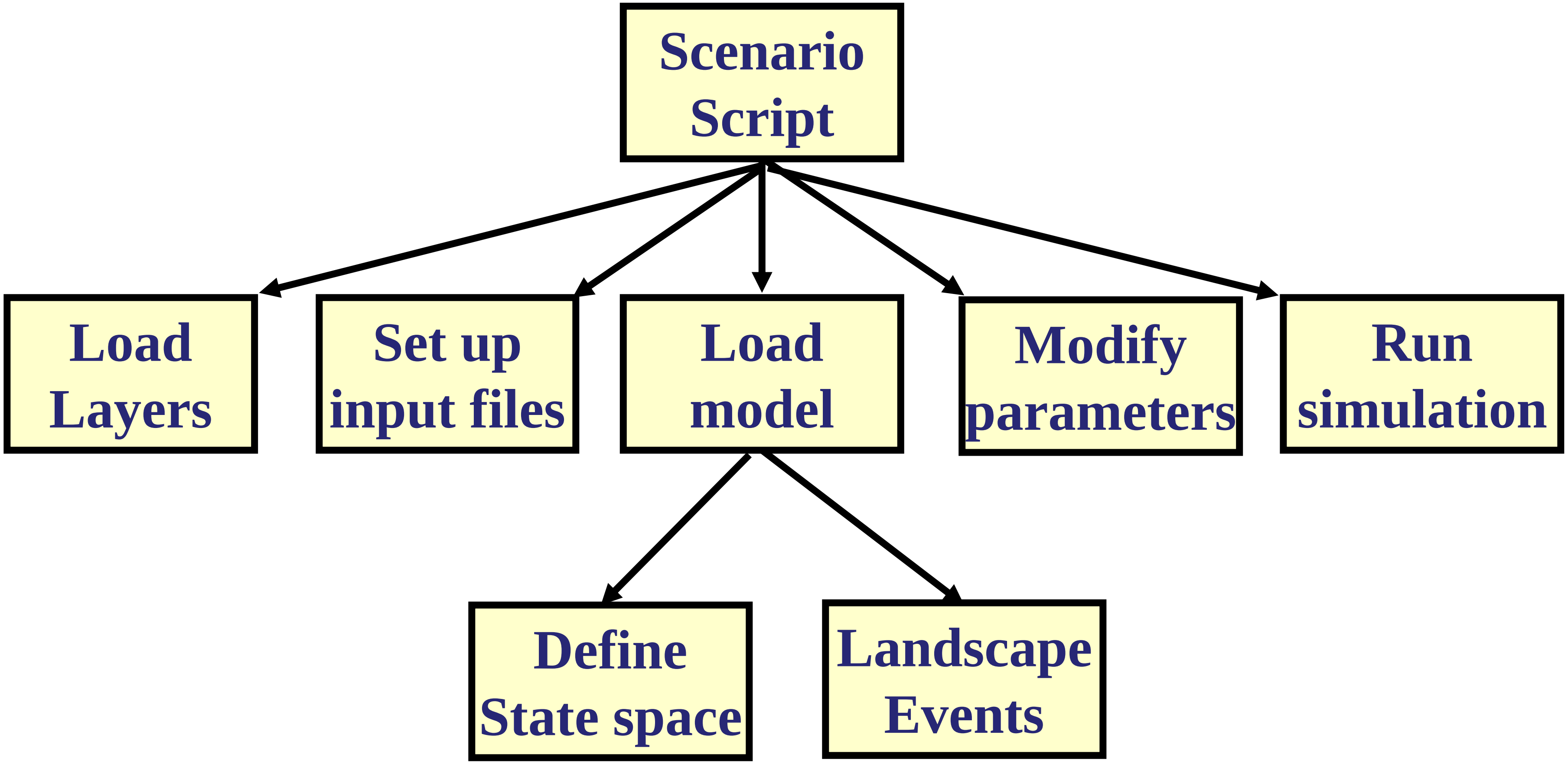
- How do we know what parameters a model has?
  - What is a parameter?
    - global variables, input files, input layers
  - Documentation and user interface
    - good to show what aspects developer wanted to you to see

- Without delving into scenario scripts
  - models will be complete black boxes
  - very limited ability to apply models or adapt to new areas

➢ Understanding the scenario language is a prerequisite for a driver’s license!

Scenario
Scripting Language
(.scn files)

SELES Scenario Structure



Typical Script Structure

• set up *script* variables	x = 25
• load required input layers	StudyArea = …
• set model dimensions	Model Dimensions: StudyArea
• load model config (.sel) file	Fire.sel
• modify parameter values	Rotation = 100
• move to defined output folder	cwd ..\output
• modify display state of views	Minimize Static
• run simulation(s)	SimStart 100 1

SELES Scripting Language

basics

- Procedural: step by step sequence of *script commands*

- *Scripts* are used to manage and run SELES models
  - ➢ Simple scripts may just load and run a model, but complex script may run sequences of models or iterative experiments

- For this module: the focus is on commonly used script command types
  - ➢ see the User Documentation Part 3 section 2 for a full list of command types

Scenario Scripts

general

- Generally case insensitive (for keywords)
- First line must be:

```
Seles Scenario
```

- Last line must be blank
- If a simulation is running, some commands will *block* until it terminates
  - ➢e.g. a layer used by a simulation cannot be closed until the simulation completes

Comments

Scripts should be documented with comments

Line Comments:

```
// this is a comment
```

Long Comments

```
/* multi-line
 comment
*/
```

Raster Layers

- SELES currently supports GeoTIFF, GRASS, ERDAS, ARC ASCII and (mostly) ARC binary (.adf) formats
  - ➢ **GeoTiff is the preferred format**

- A model has one resolution and extent, so all rasters must have the same dimensions
  - – like a “layer cake”
  - – may need to align, resize and/or rescale rasters

Basic Script Commands

Loading rasters	Example:	Age = age_prj.tif
Saving rasters		Save Age grids\a1.tif Geotiff
Closing views		Close Age
Managing view displays		Tile
Setting model dimensions		Model Dimensions: Age
Changing working directory		cwd ..\Outputs
Creating folders		mkdir Outputs

➢ See User Documentation Part 3 section 2 for more syntax details

Loading Rasters

```
<Filename>

<Viewname> = <Filename>
```

Example:

```
DEM = gisData\grids\Elevation.tif
```

Note: commands in grey are not preferred or rarely used

Loading Real-value Rasters as Fixed Precision Integer Rasters

<Filename> * #Multiplier

<Viewname> = <Filename> * #Multiplier

- multiplies cell values as they are read

➢ Rasters can also have floating point representation, but we generally recommend avoiding that if possible (e.g. they are hard to display)

Example:

```
siteIndex10 = grids\SiteIndex.tif * 10
```

Saving Rasters

```
Save <ViewName> <FileName> <Type>
```

Types:

- GEOTIFF
- GRASS COMPRESSED
- GRASS
- ARC ASCII
- ERDAS8
- ERDAS16

Example:

```
Save DEM gisdata\cell\Elevation GRASS
Save DEM grids\Elevation.tif GeoTiff
```

Closing Views

Close All

Close <Viewname>

Example:

Close DEM

Managing Raster View Display

Minimize All

Minimize <viewname>

Minimize Initial State

Minimize Static

Minimize

Tile

Scenario Dimensions

Model Dimensions: #NumRows, #NumCols

Model Dimensions: <ViewName>

Example:

Model dimensions: MgmtUnit

Change Working Directory

`cwd <directory>` (or `cd <directory>`)

- Change current working directory
- Will create directory if not present

Example:

```
cwd ..\oOutput\baseCase
```

Creating Folders

```
mkdir <directory>
```

– create directory if not present

Example:

```
mkdir grids
```

Script Commands to Load and Run Models and Set Parameters

Loading a model	Example:	FireModelTopDown.sel
Running a simulation		SimStart 100
Changing global variable parameter values		MeanFireSize = 150

Note: use script variables to change the names of input tables

- ➢ See User Documentation Part 3 section 2 for more syntax details

Loading a Dynamic Model

<ModelName.sel>

- if a model configuration file was previously loaded, it will be cleared (this is sometimes done when a script loads and runs a sequence of models)

Example:

STSM.sel

Command Ordering

- **BEFORE loading a model config (.sel) file:**
  - Load rasters (initial conditions)
  - Set up input files (files to load by the .sel file)

- **AFTER loading a model config (.sel) file:**
  - Change parameter settings from defaults in .sel file
  - Change to output directory
  - Run simulation

Simulation Control

```
SimStart #RunLength
SimStart #RunLength #Runs
SimStart #RunLength #Runs Priority
```

- A model should be loaded first

Example:

```
SimStart 1000 // run once for 1000 time steps

SimStart 100 10 Low Priority
```

Setting Parameters

```
<variable> = Expression
```

Variable must exist in loaded state space

This will change the default value set when the variable was created (in a .sel file)

- Change parameters *after* loading a model but *before* running it (i.e. between loading a .sel file and a SimStart command)

Example:

```
FireRotation = 100
```

Expressions

Expr = #Value

Expr = Expr + Expr

Expr = Expr - Expr

Expr = Expr * Expr

Expr = Expr / Expr

Expr = Expr ^ Expr

Expr = Expr % Expr

Expr = (Expr)

- use parentheses to be explicit and clear

Script Variables

what they are

Script variables are "*placeholder*" variables that:

- Can be assigned text or numeric values (no typing)
- When used, they are replaced by their values as if the value was written

➢ If used where a number is expected the value will be treated like a number

➢ If used where text is expected the value will be treated like text

Script variables provide a general and powerful tool to manage scenarios

➢ For example, a script variable an be used to set a parameter value *as well as* form part of the output folder name

Script Variables

defining

Script variables are enclosed in dollar signs, and created when first assigned (can be modified after)

```
$VarName$ = “value”
$VarName$ = value
$VarName$ = #globalVar#
$VarName$ = <script var expression>
```

Examples:

```
$threshold$ = 10
$outputDir$ = “..\outputs”
```

Script Variables

usage

Put anywhere in an expression except in quoted text

When the script command is executed, the script variable will be replaced by its value

Examples:

```
cwd   $outputDir$\v1
param1 = $threshold$ + 10
```

Script Variables

applications

(a) To manage directories

Example:

```
$scn$ = BaseCase
$gisData$ = ..\..\gisData\grids
$outputDir$ = ..\..\oOutput\$scn$

initialAge = $gisData$\age_prj.tif
…
cwd $outputDir$
```

Script Variables

applications

(b) To redirect model input (virtual copy)

Example:

```
$HarvestFile$ = HarvestTarget7.txt
MyModel.sel
```

- assuming MyModel.sel uses $HarvestFile$ to load an input file (covered in Module 5)

Script Control Commands

Condition (if) commands	Example:	if (MeanFireSize> 0) …
Iteration		while (n > 0) …

- ➢ See User Documentation Part 3 section 2 for more syntax details

Conditions

```
if (condition)
    … (any commands)
end
```

Example:

```
if (Timestep EQ 100)
    disturbanceRotation = 100
end
```

Can also include an “ELSE” section

Iteration

```
while(condition)
    … (any commands)
end
```

Example:

```
while(difference > 0.1)
    …. // change parameters
    SimStart 100 // run simulation
end
```

Iteration

over integer sequences

```
for ($var$ = #StartNumber : #EndNumber)
    … (any commands)
end
```

Default increment is 1

Must be run using a script variable

Example:

```
for($x$ = 1:5)
            param1 = $x$
            …
end
```

Iteration

with larger step increments

```
for($var$ = #Number : #Number, #Step)
    … (any commands)
end
```

Example:

```
for($x$ = 0: 100,10) // use increments if 10 from 0 to 100
```

Iteration

over file names

```
for($var$ = “filenameExpr”)
    … (any commands)
end
```

Example:

```
for($x$ = \outputRasters\ageClass*)
ac = \outputRasters\ageClass$x$
…
```

- The asterisk (“*wildcard*”) represents the portion of a filename to match (there must be at least one wildcard
- If there is a single wildcard ‘*’: $x$ will sequentially take on labels that match just the wildcard
- If there are multiple wildcards: $x$ will sequentially take on entire file names that match

Advanced and Miscellaneous Commands

Sub-scenario scripts	Example:	loadBaseLayers.scn
Scheduling commands		schedule($reportTime$) …
System commands		system “copy a.txt aBak.txt

- See User Documentation Part 3 section 2 for more syntax details

Sub-scenario scripts

Scenario: <subScenario.scn>

- ➢ loads a sub-scenario script as if it was written in the calling script
  - Note 1: cannot use script variables for sub-scenario name (but can use “if” statements to load different sub-scenarios)
  - Note 2: path is relative to the directory of main scenario

Examples:

```
Scenario: loadBaseLayers.scn
Scenario: defaultParameters.scn
```

Scheduling Commands

```
schedule(timestep)
   … (any commands)
end
```

Useful to schedule changes in inputs or outputs at certain time points

Example:

```
schedule(10)
        $x$ = #year#
        waterLevel = grids\waterLevel$x$.tif
end
```

System Commands

```
system "command“
```

- mostly used to delete, copy and rename files

- should be avoided if possible (use script variables to change names of input files rather than copying)

Example:

```
system "copy AAC1.txt AAC.txt"
```

Directories

how to know how files relate

- Starting directory for processing a scenario:
  - Directory of the scenario file
- Ending directory after scenario processed
  - Current working directory
- Directory for files loaded in .sel file:
  - Relative to the directory of the .sel file
- Directory of output during a simulation:
  - Current working directory

Hands-on

automating simulation

Start the LSEditor and open Scenarios\FireTopDown.scn

- The commands are:
  a) Load the spatial inputs (studyArea.tif and initialTSF1.tif) – these are 500 row x 500 col grids with a resolution of 1 ha.
  b) Set the model dimensions using the StudyArea layer
  c) Load the model configuration FireModelTopDown.sel file
  d) Minimize some layers and tile views

- Add the following command at the end:

```
SimStart 2000
```

➢ **Note: the last line of .scn files must be blank (check if there are errors)**

In SELES, re-open the FireTopDown.scn scenario script, and the simulation should start automatically.

Hands-on

changing parameters in a scenario script

In the LSEditor modify FireTopDown.scn

- Add the following commands *after* loading the .sel file and *before* the SimStart command (i.e. after the global variables are created but before running):

```
MeanFiresPerYear = 10

MeanFireSize = 100
```

  - ➢ This has the same fire cycle as the default mean of 1 fire/year and mean fire size of 1000 ha

In SELES, re-open the FireTopDown.scn scenario script, and the simulation should start automatically with the revised parameters.

Hands-on

adapt the model to the case study

In the LSEditor modify FireTopDown.scn (make a copy)

- Change the input layers to use the ones from the case study (which should be in a sibling folder in the main models folder):

  StudyArea = ..\..\CaseStudy\gisData\grids\studyArea.tif

  initialTimeSinceFire = ..\..\CaseStudy\gisData\grids\zero.tif

  - "..\.." goes up two levels from the Scenarios folder to the CaseStudy folder,
  - "CaseStudy\gisData\grids" is the path from the models folder to the case study grids
  - The case study has a studyArea.tif GeoTiff file, and the zero.tif GeoTiff can be used for the initial time since fire (all 0's)

Note: loading inputs can be made more elegant and robust by using script variables (e.g. by creating a $gisData$ script variable to store the common path)

In SELES, re-open the FireTopDown.scn scenario script, and the simulation should start automatically using the inputs from the case study. Note that it takes a bit for sufficient aging to be able to see fires.

Hands-on

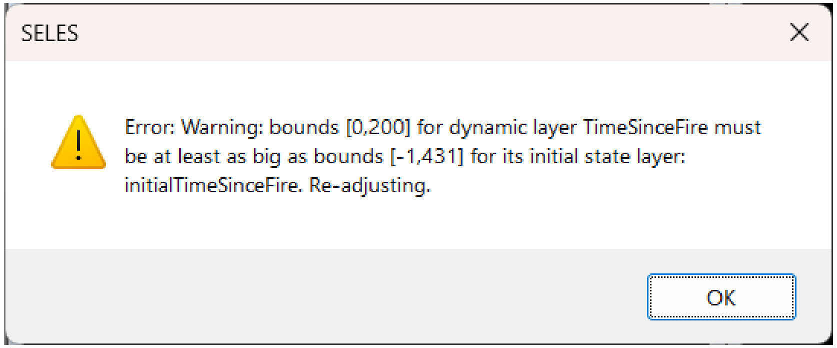
adapt the model to the case study (input compatibility)

In the LSEditor modify FireTopDown.scn

- Change the initialTimeSinceFire input layer to use the age layer :

  initialTimeSinceFire = ..\..\CaseStudy\gisData\grids\age_prj.tif

In SELES, re-open the FireTopDown.scn scenario script, and SELES issues a warning:



Error: Warning: bounds [0,200] for dynamic layer TimeSinceFire must be at least as big as bounds [-1,431] for its initial state layer: initialTimeSinceFire. Re-adjusting.

➢ The age_prj.tif raster has a range from -1 to 431 but the TimeSinceFire model layer is set to have a range from 0 to 200

This can be addressed in a three ways:

a) Ignore it (not recommended): Pressing OK allows the model to run (but the problem persists)

b) Load a different input (e.g. create and load a new layer that is limited to the range 0 to 200 (***Exercise: apply this solution using the tools from this module***)

c) Revise the model to be more general to better support adaptability (a topic for Module 5)

Notes on Adaptability

- Models can and should be designed to be adaptable
  - However, not all potential pitfalls may be foreseen (so use caution when adapting models to new study area)
  - The issue on the preceding hands-on was designed to be trivial for illustration, but some pitfalls may be very subtle

- To support adaptability, models should be well documented, in particular regarding the required inputs
  - The art of modelling in SELES will be a topic of subsequent module