Code structure of SoilGen3.8

For the description of the functionality of SoilGen and how to work with the model via the User Interface, I refer to the eBook publication https://doi.org/10.1007/978-3-031-55583-1 available via Springer.
The code is developed in (or has been migrated from Fortran to) Lazarus FreePascal, and most of the procedures and functions have internal documentation describing the order and method of actions. I used Lazarus version 2.2.6 but updated versions are released frequently.

The main program consists of various windows (programmed by *forms*) that can be opened via the user interface. The behavior of these forms and their components (e.g. buttons) is coded in *units*, but also units exist that are not strictly related to a form (-component). Names of the units always start with *unt**.
The Integrated Debugger Environment by Lazarus for SoilGen3.8 will open when double-clicking **SoilGen.inf.** The window *Source Editor* will depict the source code of each one of the units. After setting the cursor to one of the units associated with a form, the form can be shown by pressing the F12 or the associated icon on the control bar.

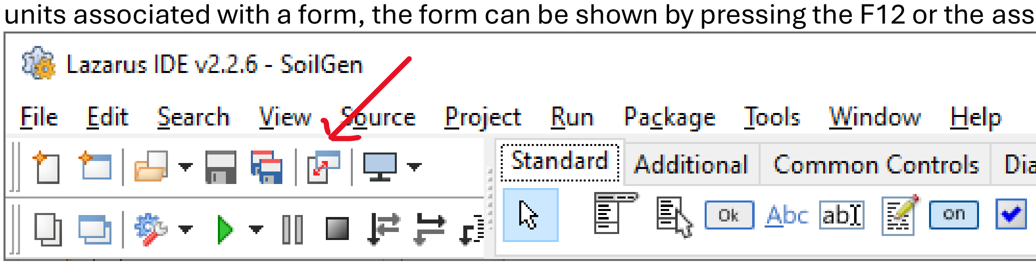


Figure 1 names the main units.

- The units in the pink marked part of this figure correspond to forms related to defining the initial situation of a model run.
- The units in the orange part correspond to windows that are used to define the scenario. Central is the unit *untGUI*, the window that opens when the program is started and from which the model is actually run. The button *bttnRunClick* on *untGUI* starts the actual simulation.

- The units in the blue part relate to model execution. *untMemDump* opens a window to depict the status of model variables in case of a runtime error. *untProcesses* contains all the processes that affect the soil at a temporal resolution of less than 1 year, and this unit is called annually by *untGui*. Processes with a temporal resolution of 1 year are part of *untGUI*.

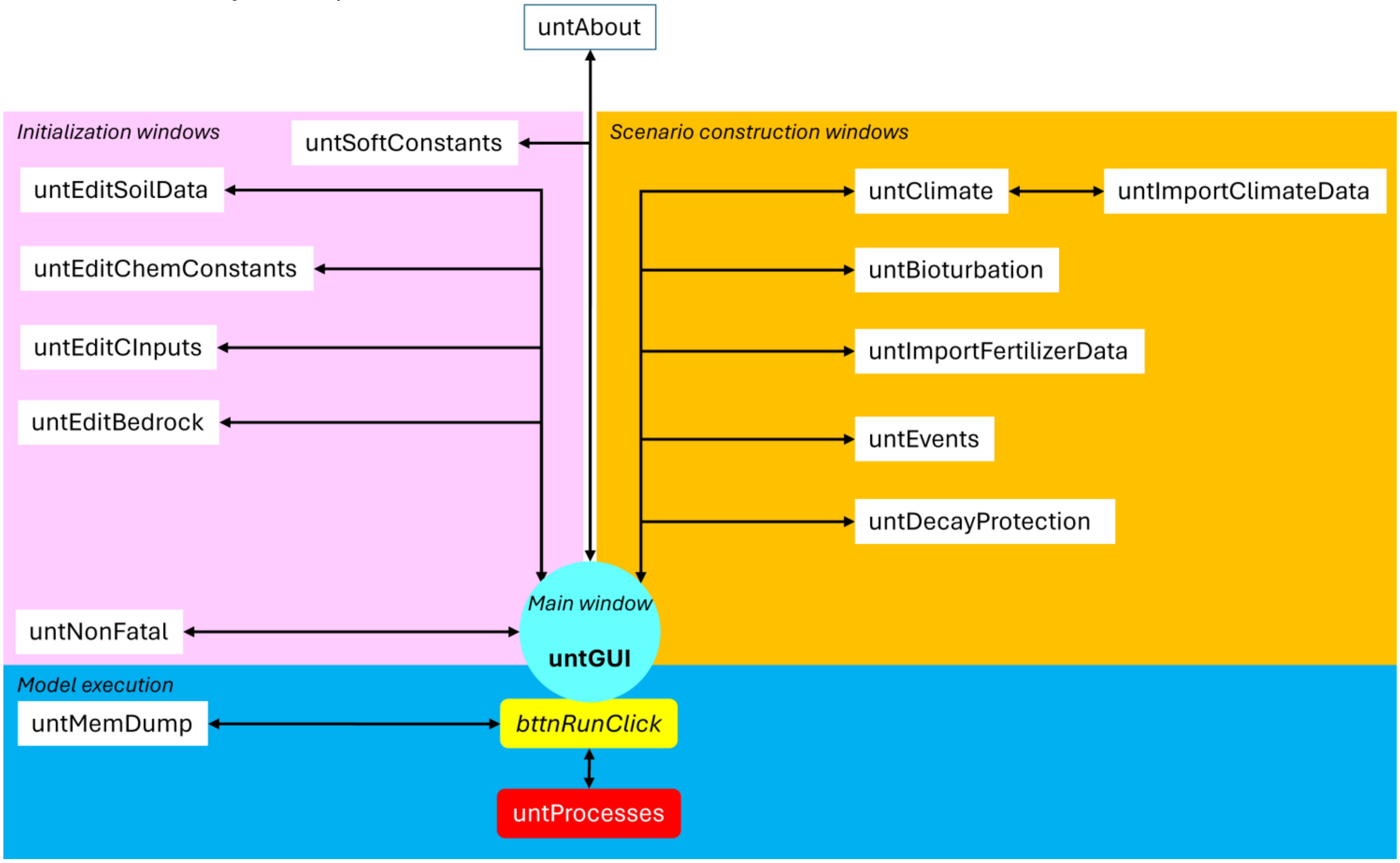
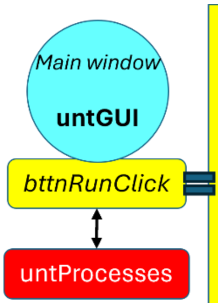


Figure 1 Major structure of the windows and associated units

Figure 2 names the functional steps that are taken when the button *bttnRun* is clicked (the *onClick* event). Several of these steps are associated with calls to functions or procedures from the unit *untGUI*. When all data for a particular year of the simulation are prepared, a call to the procedure StartProcesses in the unit *untProcesses* is done, which starts the calculations for timesteps inside the year. After that year, simulated data are passed back (by *untProcesses.*PassDataOut) to *untGUI.bttnRunClick* and a run for the next year is prepared and started.



1. ***Conditionally***: Check if run=continuation and then read soil variables; Check if capacity flow and then set some parameters
2. Set presence of read VGN params to false
3. Write header to 'PSD.txt' (particle size distribution top compartment)
4. Disable buttons in user interface
5. ***Conditionally***: exclude bioturbation, climate change, fertilization, events depending on GUI-settings
6. call ReadCheckClimate (interpolate climate input to annual climate values)
7. Initialize physical weathering
8. ***Conditionally***: If bedrock then initialize grainsize data in bedrock layer
9. call ReadCheckFertil (construct annual fertilization data from input)+call ReadCheckEvents (construct annual events data from input)+call Beschrijf_scenario (write scenario to logfile)
10. Write header to 'ectorganic.out'
11. Create time depth diagram files (*.tdd) and write headers.
12. Initialize 14C activity data
13. call Screenoutput0 (initialize screen with 6 tdd graphs)
14. call CopyRecToFrom (set all data in record DataYearIn equal to DataInit)
15. Couple bioturbation, climate, fertilization and events files, if any.
16. call CalcSlopeEffects (to get correction factors for P and PE based on slope, wind, latitude)
17. ***Conditionally***: call ReadDataRock (bedrock mineral props) + convert to mass + calculate rock porosity + initialize grain size fractions. Create 'Bedrock.out' + write initial mass per mineral
18. call CopyRecToFrom (set all data in record DataYearIn equal to DataInit)

FOR EACH YEAR

19. ***Conditionally***: if daily output tdd's are requested then call AssignDailyTDD
20. ***Conditionally:*** if tracing option applies for this year then set an indicator variable
21. ***Conditionally:*** if the program abortion checkbox was checked then call WriteData, call WriteDataC, call WriteDataWE, write logfile, raise EInOutError
22. Assign pCO2 for the current year
23. Read climate data for this year (or for 12 months in this year; if special option chckbxSpecialClimate activated)
24. Calculate correction factor for P and PE data in default year using the actual values from the climate file
25. Calculate correction factor for P and PE for slope, wind, latitude
26. Convert daily temperature data in default year using the actual values from the climate file
27. Correct daily P with interception evaporation, which is vegetation dependent and the correction factor
28. Correct weekly PE for for slope, wind, latitude
29. ***Conditionally***: If chckbxSpecialClimate activated: correct monthly P for interception + monthly P and PE with correction factor
30. Read fertilization data for this year (if any)
31. Read events for this year (if any) and read per type of event the descriptive data. Put data in DataInit, DataYearIn and DataYearOut after initializations
32. Set plant growth data for this year, can be "constants" for some vegetations
33. call WriteDataC, WriteDataWE, WriteData (inputfiles for this year)
34. Calculate and report to the GUI some progress variabes
35. Start the run inside the year (untProcesses.StartProcesses)
36. ***Conditionally***: If bedrock then write mass per mineral and rockporosity and weathering loss at end of current year
37. ***Conditionally:*** if daily output tdd's are requested then call CloseDailyTDD
38. Call ReadOutC, Call ReadOutWE (yearly outputs for C and minerals)
39. Update mass per textural class
40. call ReCalcOC (from mass per OC-pool to over-all OC mass, also include effect of physical weathering)
41. call AdjustCECAXSE2OC (adjust the CEC to the changed OC-content)
42. call Turbation (apply bioturbation, if any, to the indicated toplayers)
43. ***Conditionally***: if agriculture then call Turbation to emulate harrowing; if set: first apply turning plow
44. Update 'ectorganic.out' at the end of this year
45. call CopyRecToFrom (copy record DataYearOut to DataYearIn, for next simulation year) + call
46. call EffectSaltLoss (CaCO3 and CaSO4 mass-loss/-gain on texture) + call WeatheringIndices + call Decay (14C decay)
47. ***Conditionally***: If bedrock and rockporosity>=threshold and weatheringloss>=threshold then convert bedrock compartment to saprolite (=soil) compartment
48. with derived soil data. call CopyRecToFrom (copy record DataYearOut to DataYearIn, for next simulation year)
49. call ScreenOutput1 (update screen with 6 tdd graphs)
50. call WriteTimeDepth (write soil data at end of year to *.tdd-files)

51. Write final Van Genuchten parameters to file
52. call Prepare_Continuation (create subfolder 'continuation' and put in all the necessary files for a subsequent SoilGen run)
53. call WriteData, call WriteDataC, call WriteDataWE (inputfiles for a future year)
54. write closing remarks to logfile + Close all outputfiles and call cleanupdisk (remove temporary files) + Make a batch file of the current run if that did not exist yet + Handle runtime errors.

Figure 2 Functionality inside the bttnRunClick (i.e. the driver for a multi-annual run). Blue highlighted are the calls to procedures in the untGUI unit. Red highlighted is the call to the entry point of the unit untProcesses (the procedure StartProcesses). Numbers 1..54 also occur as remarks { 1 } (etc.) at the starting point of the relevant part of the source code of bttnRunClick.

Figure 3 describes the main structure of the unit *untProcesses*, which handles the calculations inside one year. The entry point is the procedure *StartProcesses*, which is the main process driver. Starting with initializations, calls are made to *MAINC* and *MAINC2*; the latter routine handles the calculations inside the year, with dynamic timesteps depending on the process dynamics of (a.o.) water infiltration (procedure *TSTEP*). The largest timestep is still a fraction of one day, the smaller timesteps are in the order of minutes. After one year has passed, the routine *STOPC* is called which closes output files and gives focus back to *bttnRunClick* in the user interface (*untGUI*).

The supporting procedures and functions called from *untGUI.bttnRunClick* (Figure 2) and from the process driver in *untProcesses* (Figure 3) have internal documentation in the code which is not repeated here for brevity.

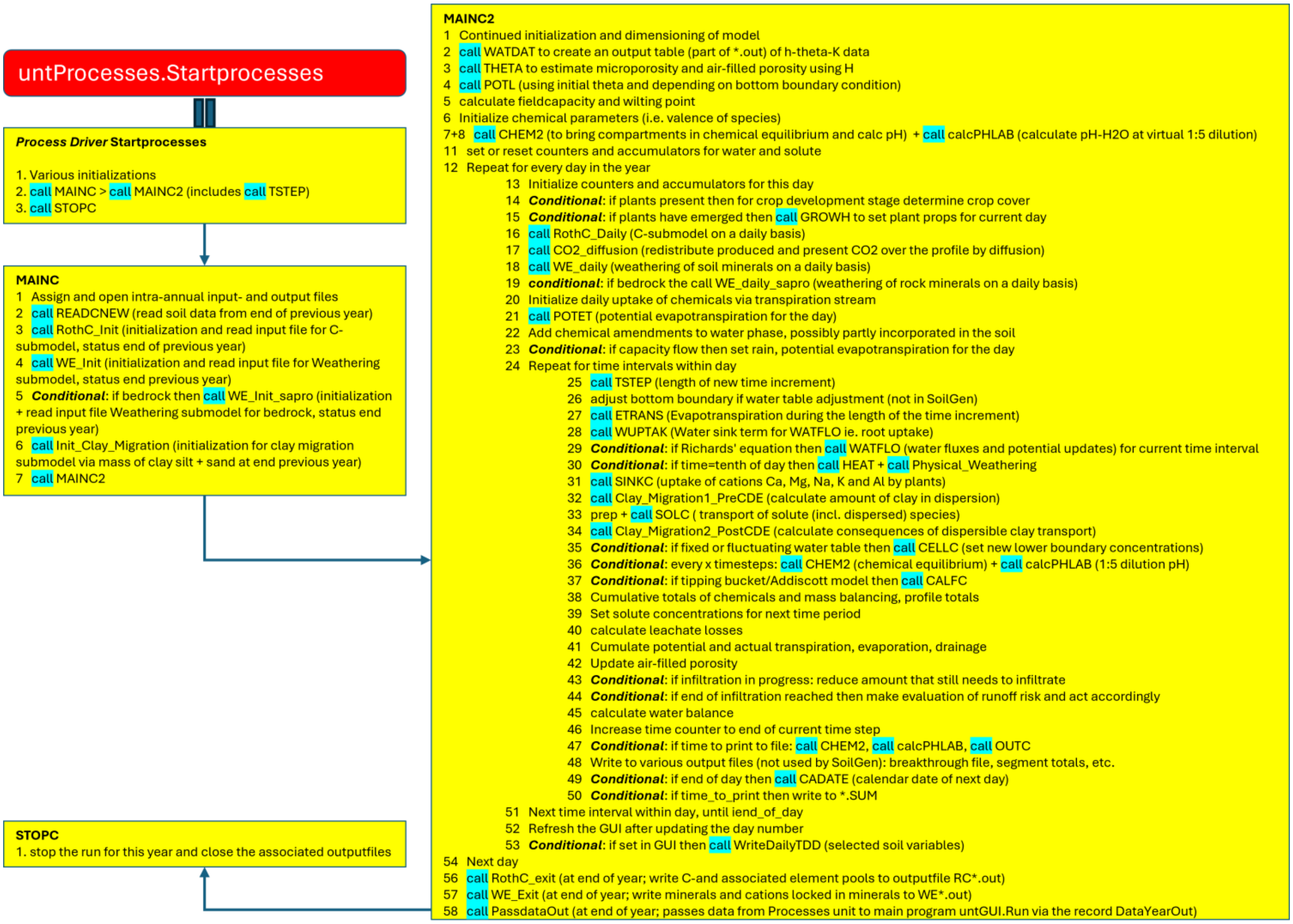


Figure 3 Structure of the process driver in untProcesses, the unit that performs the calculations inside each simulation year. Highlighted in blue are calls to supporting routines. The numbers are also indicated in the code at the starting point of the calculations. A fixed format applies, e.g. the first process "Various Initializations" in "Startprocesses" would appear as the comment {StartProcesses 1 } {process driver | Various initializations} in the model code.