```
breed [ hives hive ]
breed [ eggCohorts eggCohort]
breed [ larvaeCohorts larvaeCohort]
breed [ pupaeCohorts pupaeCohort]
breed [ IHbeeCohorts IHbeeCohort]  ; in-hive bee
breed [ droneEggCohorts droneEggCohort]
breed [ droneLarvaeCohorts droneLarvaeCohort]
breed [ dronePupaeCohorts dronePupaeCohort]
breed [ droneCohorts droneCohort]
breed [ foragerSquadrons foragerSquadron ]
  ; small group of foragers, groupsize: SQUADRON_SIZE
breed [ miteOrganisers miteOrganiser ]
  ; keep track of mites in brood cells
breed [ flowerPatches flowerPatch]
breed [ Signs Sign ]
  ; signs to inform the user

globals [
 ABANDON_POLLEN_PATCH_PROB_PER_S
 AFF
 AFF_BASE
 AllDaysAllPatchesList
 BugAlarm
 ColonyDied
 ColonyTripDurationSum
 ColonyTripForagersSum
 CROPVOLUME
 CumulativeHoneyConsumption
 DailyForagingPeriod
 DailyHoneyConsumption
 DailyMiteFall
 DailyPollenConsumption_g
 Day
 DeathsAdultWorkers_t
 DeathsForagingToday
 DecentHoneyEnergyStore
 DRONE_EGGLAYING_START
 DRONE_EGGLAYING_STOP
 DRONE_EMERGING_AGE
 DRONE_HATCHING_AGE
 DRONE_LIFESPAN
 DRONE_PUPATION_AGE
 DRONE_EGGS_PROPORTION
 EMERGING_AGE
 EmptyFlightsToday
 ENERGY_HONEY_per_g
 ENERGY_SUCROSE
 ExcessBrood
 FIND_DANCED_PATCH_PROB
 FLIGHT_VELOCITY
```

FLIGHTCOSTS_PER_m
FORAGER_NURSING_CONTRIBUTION
FORAGING_STOP_PROB
ForagingRounds
ForagingSpontaneousProb
HarvestedHoney_kg
HATCHING_AGE
HONEY_STORE_INIT
HoneyEnergyStore
HoneyEnergyStoreYesterday
HoPoMo_seasont
IdealPollenStore_g
InhivebeesDiedToday
INVADING_DRONE_CELLS_AGE
INVADING_WORKER_CELLS_AGE
InvadingMitesDroneCellsReal
 ; actual number of mites invading the cells, might be
 ; lower than theor. number, if brood cells are crowded with mites
InvadingMitesDroneCellsTheo
 ; theoretical number of mites invading the cells
InvadingMitesWorkerCellsReal
InvadingMitesWorkerCellsTheo
LIFESPAN
LostBroodToday
 ; brood that die due to lack of nursing or lack of pollen today
LostBroodTotal ; .. and summed up
MAX_AFF
MAX_BROOD_NURSE_RATIO
MAX_DANCE_CIRCUITS
MAX_EGG_LAYING
MAX_HONEY_ENERGY_STORE
MAX_INVADED_MITES_DRONECELL
MAX_INVADED_MITES_WORKERCELL
MAX_PROPORTION_POLLEN_FORAGERS
MAX_TOTAL_KM
MIN_AFF
MIN_IDEAL_POLLEN_STORE
MITE_FALL_DRONECELL
MITE_FALL_WORKERCELL
MITE_MORTALITY_BROODPERIOD
MITE_MORTALITY_WINTER
MORTALITY_DRONE_EGGS
MORTALITY_DRONE_LARVAE
MORTALITY_DRONE_PUPAE
MORTALITY_DRONES
MORTALITY_DRONES_INFECTED_AS_PUPAE
MORTALITY_EGGS
MORTALITY_FOR_PER_SEC
MORTALITY_INHIVE
MORTALITY_INHIVE_INFECTED_AS_ADULT
MORTALITY_INHIVE_INFECTED_AS_PUPA

MORTALITY_LARVAE
MORTALITY_PUPAE
N_FLOWERPATCHES
N_GENERIC_PLOTS
NectarFlightsToday
NewDroneEggs
NewDroneLarvae
NewDronePupae
NewDrones
NewDrones_healthy
NewForagerSquadronsHealthy
NewForagerSquadronsInfectedAsAdults
NewForagerSquadronsInfectedAsPupae
NewIHbees
NewIHbees_healthy
NewReleasedMitesToday
 ; all (healthy and infected) mites released from cells (mothers+offspring)
 ; on current day (calculated after MiteFall!)
NewWorkerEggs
NewWorkerLarvae
NewWorkerPupae
PATCHCOLOR
PhoreticMites ; all phoretic mites, healthy and infected
PhoreticMitesHealthyRate
POLLEN_DANCE_FOLLOWERS
POLLEN_STORE_INIT
PollenFlightsToday
POLLENLOAD
PollenStore_g
PollenStore_g_Yesterday
POST_SWARMING_PERIOD
PRE_SWARMING_PERIOD
ProbPollenCollection
PropNewToAllPhorMites
PROTEIN_STORE_NURSES_d
ProteinFactorNurses
Pupae_W&D_KilledByVirusToday
 ; number of drone + worker pupae that were killed by the virus today
PUPATION_AGE
Queenage
RecruitedFlightsToday
SaveInvadedMODroneLarvaeToPupae
SaveInvadedMOWorkerLarvaeToPupae
SaveWhoDroneLarvaeToPupae
SaveWhoWorkerLarvaeToPupae
SEARCH_LENGTH_M
SearchingFlightsToday
SEASON_START ; defines beginning of foraging period
SEASON_STOP ; end of foraging period & latest end of drone production
SimpleDancing
STEPWIDTH

```
 STEPWIDTHdrones
 SumLifeSpanAdultWorkers_t
 SummedForagerSquadronsOverTime
 SwarmingDate
 TIME_UNLOADING
 TIME_UNLOADING_POLLEN
 TodaysAllPatchesList
 TodaysSinglePatchList
 TotalBeesAdded
   ; beekeeper can add bees in autumn, these are added up as long
   ; as simulation runs
 TotalDroneEggs
 TotalDroneLarvae
 TotalDronePupae
 TotalDrones
 TotalEggs
 TotalEventsToday        ; sum of todays "xxxFlightsToday"
 TotalForagers
 TotalFPdetectionProb
 TotalHoneyFed_kg
   ; if "beekeeper" has to feed the colony, fed honey is added up as long
   ; as simulation runs
 TotalHoneyHarvested_kg
 TotalIHbees
 TotalLarvae
 TotalMites
 TotalPollenAdded
   ; beekeeper can add pollen in spring, which is added up as long
   ; as simulation runs
 TotalPupae
 TotalWeightBees_kg ; weight of all bees (brood, adults, drones..)
 TotalWorkerAndDroneBrood
 VIRUS_KILLS_PUPA_PROB
 VIRUS_TRANSMISSION_RATE_MITE_TO_PUPA
   ; probability for an infected invaded mite to infect the bee pupa
 VIRUS_TRANSMISSION_RATE_PUPA_TO_MITES
   ; probability for an infected bee pupa to infect healthy invaded mites
 WEIGHT_WORKER_g


    AllBeeMappCorrectionsList   ;   ***NEW FOR BEEHAVE_BEEMAPP2015***
    AssessmentNumber            ;   ***NEW FOR BEEHAVE_BEEMAPP2015***
    WeatherDataList             ;   ***NEW FOR BEEHAVE_BEEMAPP2015***


]

turtles-own ; all cohorts below have these variables too
[
  age
```

```
  ploidy
  number
  numberDied
  invadedByMiteOrganiserID
]

pupaeCohorts-own
[
  number_infectedAsPupa
  number_healthy
]

dronePupaeCohorts-own
[
  number_infectedAsPupa
  number_healthy
]

IHbeeCohorts-own
[
  number_infectedAsPupa
  number_infectedAsAdult
  number_healthy
  ]

droneCohorts-own
[
  number_infectedAsPupa
  number_healthy
]

foragerSquadrons-own
[
  activity
  activityList
  knownNectarPatch
  knownPollenPatch
  pollenForager
  cropEnergyLoad
  collectedPollen
  mileometer
  km_today
  infectionState
]

flowerPatches-own
[
  patchType
  distanceToColony
  xcorMap
  ycorMap
```

```
  oldPatchID
  size_sqm
  quantityMyl
  amountPollen_g
  nectarConcFlowerPatch
  detectionProbability
  flightCostsNectar
  flightCostsPollen
  EEF
  danceCircuits
  danceFollowersNectar
  summedVisitors
  nectarVisitsToday
  pollenVisitsToday
  tripDuration
  tripDurationPollen
  mortalityRisk
  mortalityRiskPollen
  handlingTimeNectar
  handlingTimePollen
]

miteOrganisers-own
[
  workerCellListCondensed
  droneCellListCondensed
  cohortInvadedMitesSum
  invadedMitesHealthyRate
  invadedDroneCohortID
  invadedWorkerCohortID
]


; ========== BUTTONS ============================================================
; *******************************************************************************

to Setup ; BUTTON!
  clear-all
  set N_INITIAL_BEES round N_INITIAL_BEES
  set N_INITIAL_MITES_HEALTHY round N_INITIAL_MITES_HEALTHY
  set N_INITIAL_MITES_INFECTED round N_INITIAL_MITES_INFECTED
  reset-ticks
  if ReadInfile = true [ ReadFileProc ]
  ParameterizationProc
  ifelse ReadInfile = false
    [ CreateFlowerPatchesProc ]
      ; IF: flower patches are defined by input fields in GUI
    [ Create_Read-in_FlowerPatchesProc ]
      ; ELSE: or read in from a text file
  if ReadBeeMappFile = true ;                ***NEW FOR BEEHAVE_BEEMAPP2015***
    [ ReadBeeMappFileProc ]
```

```
  CreateImagesProc
  if (Experiment = "Experiment A") or (Experiment = "Experiment B")
    [
      user-message "Please make sure experimental colony conditions are defined in Setup and
GoTreatmentProc"
      ;(INSERT INITIAL CONDITIONS FOR EXPERIMENTAL COLONIES HERE)
      GoTreatmentProc
    ]




end

;
**********************************************************************************
**********************************************************************************
************

to CreateOutputFileProc
  ; BUTTON!  writes data in file, copied from:
  ; Netlogo: Library: Code Examples: Output_Example.nlogo

  set WriteFile true
  let filename "Output.txt"
  if is-string? filename  ; to make sure filename is a string
  [
    if file-exists? filename   ; if the file already exists, it is deleted
    [
      file-delete filename
    ]
    file-open filename
    WriteToFileProc  ; record the initial turtle data
  ]
end

;
**********************************************************************************
**********************************************************************************
************

to StartProc
  ; called by Day/Month/Year/x days and RUN Button

  if BugAlarm = true
  [
    ask patches
    [
      set pcolor red
    ]
    user-message ("BUG ALARM!! (Start Proc)")  stop
```

```
    ; programm is stopped, if an "assertion" is violated, background becomes red
  ]

  if (stopDead = true) and (ColonyDied = true) [ stop ]
    ; programm is stopped, if colony is dead and stopDead switch is "On"

  Go ; <<<<<<<<<<<<<<<<<<<<<<<<<<

  if WriteFile = true [ WriteToFileProc ]
    ; results are recorded in Output
    ; file after each timestep

end ; StartProc

;=============================================================================
=============================================================================
=============

to ParameterizationProc


;  begin ***NEW FOR BEEHAVE_BEEMAPP2015***
 set WeatherDataList []
 if Weather = "Weather File"
 [
   ifelse file-exists? WeatherFile
   [
   file-open WeatherFile
   while [not file-at-end?]
   [
     set WeatherDataList lput read-from-string(file-read-line) WeatherDataList
   ]
   file-close
   ]
   [ user-message "No such weather input file available!" ]
 ]
;  end ***NEW FOR BEEHAVE_BEEMAPP2015***


; BROOD CARE:
  set FORAGER_NURSING_CONTRIBUTION  0.2
  set MAX_BROOD_NURSE_RATIO  3
    ; 3  (3: Free & Racey 1968) (Becher et al. 2010: 2.65)
    ; # brood that can be raised by a single "nurse" bee ("nurse": IH-bee and
    ; to some degree also foragers!, see FORAGER_NURSING_CONTRIBUTION)

; COLONY:
  set ColonyDied  FALSE
  set DRONE_EGGS_PROPORTION  0.04
    ; 0.04  Wilkinson&Smith 2002 (from Allen 1963, 1965)
```

```
  set MIN_IDEAL_POLLEN_STORE  250
   ; 250  [g] min. amount of pollen that a colony tries to store

  set POLLEN_STORE_INIT  100
   ; 100 [g] pollen present on 1st day of simulation

  set PRE_SWARMING_PERIOD  3
   ;  HoPoMo: 3d, see also Winston p. 184

  set PROTEIN_STORE_NURSES_d  7
   ;  7 [d] Crailsheim 1990

  set ProteinFactorNurses  1
   ; 0..1, is daily calculated in PollenConsumptionProc, reflects protein
   ; content of brood food

  set Queenage  230      ;  queen emerged mid of May

  set WEIGHT_WORKER_g  0.1
   ; 0.125  0.1 or 0.11 or 0.125
   ; (0.1: HoPoMo 0.11: ; Martin 1998: 1kg adults = 9000 bees)
   ; (0.125: Calis et al. 1999) higher weight => less mites!

; DEVELOPMENT:
  set AFF_BASE  21      ; like BEEPOP
  set MIN_AFF 7  ; Robinson 1992: 7d; see also: Winston 1987, p. 92/93
   ; models: Amdam & Omholt 2003, Beshers et al 2001: 7d
  set MAX_AFF 50
   ; within range given in Winston 1987, p. 92/93
  set DRONE_EGGLAYING_START  115
   ;  115: 25.April (Allen 1963: late April ..late August)
  set DRONE_EGGLAYING_STOP  240
   ; 240  240: 28.August (Allen 1963: late April ..late August)
  set DRONE_HATCHING_AGE  3     ; Jay 1963, Hrassnig, Crailsheim 2005
  set DRONE_PUPATION_AGE  10    ; i.e. capping of the cell; Fukuda, Ohtani 1977
  set DRONE_EMERGING_AGE  24
  set HATCHING_AGE  3          ;  Winston p. 50
  set PUPATION_AGE  9    ; i.e. capping of the cell
  set EMERGING_AGE  21
  set MAX_EGG_LAYING 1600        ; 1600  max. # eggs laid per day

; ENVIRONMENT
  set SEASON_START  1      ; season: 1st January - 31st December, i.e.
  set SEASON_STOP  365     ; foraging potentially possible throughout the year (weather depending)
  set ABANDON_POLLEN_PATCH_PROB_PER_S  0.00002

; FORAGING
  set CROPVOLUME  50
   ; 50   [microlitres] (~50mg nectar) Winston (1987), Nuñez (1966, 1970), Schmid-Hempel et al.
(1985)
  set FIND_DANCED_PATCH_PROB  0.5; (0.5 = ca. average of reported values):
```

```
  ; Seeley 1983: recruits required 4.8 dance-guided search trips to find target patch = 0.21
  ; Judd 1995: of 63 dance followers, 16 were successful, 16/63 = 0.25
  ; Biesmeijer, deVries 2001: review: 0.95 (Oettingen-Spielberg 1949), 0.73 (Lindauer 1952)

 set FLIGHT_VELOCITY  6.5
  ; 6.57084    [m/s] derived from Seeley 1994, mean velocity
  ; during foraging flight see also Ribbands p127: 12.5-14.9mph (*1.609=20.1-24.0 km/h =
  ; 5.58-6.66m/s)

 set FLIGHTCOSTS_PER_m 0.000006   ;
  ; [kJ/m] Flightcosts per m (Goller, Esch 1990: 0.000006531 kJ/m,  (assuming speed of 6.5m/s:
  ; flight costs: 0.0424W - compare with Schmid-Hempel et al. 1985: 0.0334W => 0.000005138 )

 set FORAGING_STOP_PROB  0.3

 set MAX_DANCE_CIRCUITS  117              ;  (117) (Seeley, Towne 1992)
 set MAX_PROPORTION_POLLEN_FORAGERS  0.8      ;  (0.8: Lindauer 1952)
 set POLLEN_DANCE_FOLLOWERS  2     ; 2: number of bees, following a pollen dancer
 set POLLENLOAD  0.015
  ; [g] 0.015g average weight of 2 pollen pellets, HoPoMo: 15 mg: "On average,
  ; one pollen foraging flight results in 15mg of collected pollen (Seeley, 1995)"

 set ProbPollenCollection  0
  ; probability to collect pollen instead of nectar  calculated in ForagingRoundProc

 set SEARCH_LENGTH_M 17 * 60 * FLIGHT_VELOCITY  ; 17*60*6.5 = 6630m
  ; [m] distance (= 17 min!), a unsuccesful forager flies on average
  ; Seeley 1983: search trip: 17min (+-11)

 set SimpleDancing FALSE
  ; (false) if true: fixed nectar dancing TH and fixed number of dance followers

 set TIME_UNLOADING  116
  ; (116) [s] time, a nectar forager needs to become unloaded  derived from Seeley 1994

 set TIME_UNLOADING_POLLEN  210
  ; (210s = 3.5 min) [s]  Ribbands p.131: 3.5 minutes (Park 1922,1928b)

 set TotalFPdetectionProb  -1
  ; correct value is set in "Foraging_searchingProc" but only when searching takes places

; MORTALITY
 set DRONE_LIFESPAN  37
  ; Fukuda Ohtani 1977; life span drones:  summer: 14d, autumn: 32-42d
 set LIFESPAN  290
  ; [d] 290d (max. life span of worker; Sakagami, Fukuda 1968)

 set MAX_TOTAL_KM  800
  ; [ km ]  800, as mortality acts only at end of time step! 838km: max. flight
  ; performance in a foragers life (Neukirch 1982)
```

```
 set MORTALITY_DRONE_EGGS 0.064    ;  Fukuda Ohati 1977:
 set MORTALITY_DRONE_LARVAE 0.044   ;  100 eggs, 82 unsealed brood, 60 sealed brood and 56
adults
 set MORTALITY_DRONE_PUPAE  0.005
 set MORTALITY_DRONES  0.05       ; Fukuda Ohati 1977: "summer", av. lifespan: 14d
 set MORTALITY_EGGS  0.03         ;  HoPoMo p. 230: 0.03
 set MORTALITY_LARVAE  0.01       ;  HoPoMo p. 230: 0.01
 set MORTALITY_PUPAE  0.001       ;  HoPoMo p. 230: 0.001
 set MORTALITY_FOR_PER_SEC  0.00001
   ; derived from Visscher&Dukas 1997 (Mort 0.036 per hour foraging)

 set MORTALITY_INHIVE  0.004;
   ; 0.0038: derived from Martin 2001 (healthy winter
   ; based on 50% mortality) (HoPoMo: MORTALITYbase: 0.01) p. 230


; PHYSICS
 set ENERGY_HONEY_per_g  12.78
   ; [kJ/g] (= [J/mg])    Wikipedia: http://www.nal.usda.gov/fnic/foodcomp/search/

 set ENERGY_SUCROSE  0.00582      ; 0.00582 [kJ/micromol]   342.3 g/mol

; PROGRAM
 set STEPWIDTH  50      ;  for graphic
 set STEPWIDTHdrones  5      ;  for graphic
 set BugAlarm  FALSE      ;
 set N_GENERIC_PLOTS 8

; VARROA
 set MITE_FALL_DRONECELL  0.2
   ;  0.2 (20%) Martin 1998  proportion of those mites emerging from
   ; worker cells, which fall from the comb and are hence considered to die.

 set MITE_FALL_WORKERCELL  0.3
   ;  0.3 (30%) Martin 1998  proportion of those mites emerging from drone
   ; cells, which fall from the comb and are hence considered to die.

 set MITE_MORTALITY_BROODPERIOD  0.006
   ; Martin 1998: 0.006;  (0.006: Fries et al 1994, Tab. 6) daily mortality of phoretic
   ; mites during brood period

 set MITE_MORTALITY_WINTER  0.002
   ; Martin 1998: 0.002;   Fries et al 1994: 0.004 (Tab. 6)
 set NewReleasedMitesToday  0
   ;  all (healthy and infected) mites released from cells (mothers+offspring)
   ; on current day (calculated after MiteFall!)

; AUXILIARY VARIABLES
 set DecentHoneyEnergyStore  N_INITIAL_BEES * 1.5 * ENERGY_HONEY_per_g
   ; re-set in every foraging round (ForagingRoundProc )
```

```
set HONEY_STORE_INIT  0.5 * MAX_HONEY_STORE_kg * 1000
  ;  [g]   (1g Honey = 124.80kJ)

set HoneyEnergyStore  (HONEY_STORE_INIT * ENERGY_HONEY_per_g)      ;  [kJ]
set IdealPollenStore_g  POLLEN_STORE_INIT
  ; [g] is calculated daily in PollenConsumptionProc

set MAX_HONEY_ENERGY_STORE  MAX_HONEY_STORE_kg * ENERGY_HONEY_per_g * 1000 ; [kJ]
set PollenStore_g  POLLEN_STORE_INIT      ;  [g]
set NewForagerSquadronsHealthy (N_INITIAL_BEES / SQUADRON_SIZE)
  ; foragers in time step 1 are all healthy

set TotalForagers  NewForagerSquadronsHealthy * SQUADRON_SIZE
  ;  has to be set here to calculate egg laying on the 1st time step

set Aff  AFF_BASE
set INVADING_DRONE_CELLS_AGE  DRONE_PUPATION_AGE - 2
  ; 2d before capping, Boot et al. 1992 (Exp. & Appl. Acarol. 16:295-301)

set INVADING_WORKER_CELLS_AGE  PUPATION_AGE - 1
  ; 1d before capping, Boot et al. 1992 (Exp. & Appl. Acarol. 16:295-301)

set PhoreticMites  N_INITIAL_MITES_HEALTHY + N_INITIAL_MITES_INFECTED
set TotalMites  PhoreticMites
set PATCHCOLOR 38 ; colour of the background
ask patches [ set pcolor PATCHCOLOR ]
if (N_INITIAL_MITES_HEALTHY + N_INITIAL_MITES_INFECTED) > 0
[
  set PhoreticMitesHealthyRate N_INITIAL_MITES_HEALTHY
     / (N_INITIAL_MITES_HEALTHY + N_INITIAL_MITES_INFECTED)
]
if RAND_SEED != 0 [ random-seed RAND_SEED ]
  ; if RAND_SEED set to 0, random numbers will differ in every run

; MITE REPRODUCTION MODELS:
if MiteReproductionModel = "Fuchs&Langenbach"
[
  set MAX_INVADED_MITES_DRONECELL 16
    ; 16 (Fuchs&Langenbach 1989) defines length of workercell, dronecell list
    ; of MiteOrganisers

  set MAX_INVADED_MITES_WORKERCELL 8
    ; (Fuchs&Langenbach 1989)
    ; defines length of workercell, dronecell list of MiteOrganisers
]

if MiteReproductionModel = "Martin"
[
  set MAX_INVADED_MITES_DRONECELL 4
    ; defines length of workercell, dronecell list of MiteOrganisers
  set MAX_INVADED_MITES_WORKERCELL 4
```

```
    ; defines length of workercell, dronecell list of MiteOrganisers
  ]

  if MiteReproductionModel = "Test"
  [
   set MAX_INVADED_MITES_DRONECELL 5
   set MAX_INVADED_MITES_WORKERCELL 5
  ]

  if MiteReproductionModel = "Martin+0"
  [
   set MAX_INVADED_MITES_DRONECELL 5
   set MAX_INVADED_MITES_WORKERCELL 5
  ]

  if MiteReproductionModel = "No Mite Reproduction"
  [
   set MAX_INVADED_MITES_DRONECELL 5
   set MAX_INVADED_MITES_WORKERCELL 5
  ]

  ; VIRUS TYPES;
  if Virus = "DWV"
  [
   set VIRUS_TRANSMISSION_RATE_MITE_TO_PUPA 0.89  ; 0.89
   set VIRUS_TRANSMISSION_RATE_PUPA_TO_MITES 1   ; 1: Martin 2001
   set VIRUS_KILLS_PUPA_PROB 0.2   ; DWV: 0.2 (Martin 2001)
   set MORTALITY_INHIVE_INFECTED_AS_PUPA 0.012; (0.0119)
    ; if pupa was infected but survived
    ; based on Martin 2001 Survivorship curve (infected, winter)
    ; calculated at: 50% mortality(=58d);

   set MORTALITY_INHIVE_INFECTED_AS_ADULT MORTALITY_INHIVE
    ;  Martin 2001: DWV infected adults become carriers with unaffected survivorship

   set MORTALITY_DRONES_INFECTED_AS_PUPAE MORTALITY_INHIVE_INFECTED_AS_PUPA *
(MORTALITY_DRONES / MORTALITY_INHIVE)
    ; NO data on drone mortality! Use same increase in mortality as for workers
  ]

  if Virus = "APV"
  [
   set VIRUS_TRANSMISSION_RATE_MITE_TO_PUPA 1
   set VIRUS_TRANSMISSION_RATE_PUPA_TO_MITES 0
    ; 0: Martin 2001 (0, as the pupae dies! - so this value doesn't matter at all!)

   set VIRUS_KILLS_PUPA_PROB 1   ; APV: 1 (Martin 2001)
   set MORTALITY_INHIVE_INFECTED_AS_PUPA 1
    ; doesn't matter, as APV infected pupae die before emergence!

   set MORTALITY_INHIVE_INFECTED_AS_ADULT 0.2
```

```
    ; (0.2: Sumpter & Martin 2004)

   set MORTALITY_DRONES_INFECTED_AS_PUPAE MORTALITY_INHIVE_INFECTED_AS_PUPA *
(MORTALITY_DRONES / MORTALITY_INHIVE)
     ; NO data on drone mortality! Use same increase in mortality as for workers
  ]

  if Virus = "benignDWV" ; like DWV but does not harm the infected bees
  [
   set VIRUS_TRANSMISSION_RATE_MITE_TO_PUPA 0.89     ; 1
   set VIRUS_TRANSMISSION_RATE_PUPA_TO_MITES 1
    ; 0: Martin 2001 (0, as the pupae dies!)
   set VIRUS_KILLS_PUPA_PROB 0  ; (benign!)
   set MORTALITY_INHIVE_INFECTED_AS_PUPA MORTALITY_INHIVE ; (benign!)
   set MORTALITY_INHIVE_INFECTED_AS_ADULT MORTALITY_INHIVE
   set MORTALITY_DRONES_INFECTED_AS_PUPAE MORTALITY_INHIVE_INFECTED_AS_PUPA
    ; NO data on drone mortality! Use worker mortality!
  ]

  if Virus = "modifiedAPV"
  [
   set VIRUS_TRANSMISSION_RATE_MITE_TO_PUPA 1     ; 1
   set VIRUS_TRANSMISSION_RATE_PUPA_TO_MITES 1   ;
   set VIRUS_KILLS_PUPA_PROB 1  ; APV: 1 (Martin 2001)
   set MORTALITY_INHIVE_INFECTED_AS_PUPA 1
    ; doesn't matter, as APV infected pupae die before emergence!

   set MORTALITY_INHIVE_INFECTED_AS_ADULT 0.2
    ; (0.2: Sumpter & Martin 2004)

   set MORTALITY_DRONES_INFECTED_AS_PUPAE MORTALITY_INHIVE_INFECTED_AS_PUPA
   ; NO data on drone mortality! Use worker mortality!
  ]

  if Virus = "TestVirus"
  [
   set VIRUS_TRANSMISSION_RATE_MITE_TO_PUPA 1  ; 0.89
   set VIRUS_TRANSMISSION_RATE_PUPA_TO_MITES 1   ; 1: Martin 2001
   set VIRUS_KILLS_PUPA_PROB 0   ; DWV: 0.2 (Martin 2001)
   set MORTALITY_INHIVE_INFECTED_AS_PUPA 0.012; (0.0119)
    ; if pupae was infected but survived; based on Martin 2001 Survivorship
    ; curve (infected, winter) calculated at 50% mortality = 58d age

   set MORTALITY_INHIVE_INFECTED_AS_ADULT MORTALITY_INHIVE
    ;  Martin 2001: DWV infected adults become carriers with unaffected survivorship

   set MORTALITY_DRONES_INFECTED_AS_PUPAE MORTALITY_INHIVE_INFECTED_AS_PUPA
    ; NO data on drone mortality! Use worker mortality!
  ]

end;
```

```
;
*******************************************************************************
*******************************************************************************
************

to CreateImagesProc
 ; "signs" are symbols in the NetLogo "World" which are used to visualize structure
 ; and dynamics of the colony/varroa model

 create-hives 1
 [
  ifelse ReadInfile = true ;
    ; true: hive placed on the left side, else: in the centre
    [ setxy -1 4.5 ]
    [ setxy 16 4.5 ]
  set size 7 set shape "beehiveDeepHive" set color brown
 ]

 create-Signs 1
 [
  setxy 16 -15
  set shape "skull"
  set size 15
  set color black
  hide-turtle
 ] ;

 create-Signs 1
 [
  setxy 40 3
  set shape "sun"
  set size 7
  set color yellow
  hide-turtle
 ] ;

 create-Signs 1
 [
  setxy 37 2
  set shape "cloud"
  set size 7
  set color grey
  hide-turtle
 ]

 create-Signs 1
 [
  setxy 38 -10
  set shape "beelarva_x2"
  set size 8
```

```
  set color white
  facexy xcor + 1 ycor + 1 ; (turned by 45deg)
  hide-turtle
]

create-Signs 1
[
  setxy 31 3
  set shape "arrow"
  set size 4
  set color green
  facexy xcor + 1 ycor
  set label (HoneyEnergyStore - HoneyEnergyStoreYesterday)
        / ( ENERGY_HONEY_per_g * 1000 )
]

create-Signs 1
[
  setxy 26 3
  set shape "arrowpollen"
  set size 4
  set color green
  facexy xcor - 1 ycor
  set label (PollenStore_g - PollenStore_g_Yesterday)
]

create-Signs 1
  ; sign for suppressed foraging i.e. if foraging prob. is set
  ; to 0 although weather is suitable for foraging
[
  setxy 36 -4
  set shape "exclamation"
  set size 3
  set color orange
  hide-turtle
]

create-Signs 1
[
  setxy 38 -18
  set shape "pete"
  set size 6
  set color white
  set label-color black
  hide-turtle
]

create-Signs 1
[
  setxy 38 -25
  set shape "honeyjar"
```

```
  set size 6
  set color white
  hide-turtle
]

create-Signs 1
[
  setxy 38 -25
  set shape "ambrosia"
  set size 6
  set color white
  hide-turtle
]
create-Signs 1
[
  setxy 42.5 -25
  set shape "pollengrain"
  set size 7
  set color yellow
  hide-turtle
]
create-Signs 1
[
  setxy 38 -31
  set shape "varroamite03"
  set size 6
  set color 33
  set heading 0
  hide-turtle
]
create-Signs 1
[
  setxy 38 -31.2
  set shape "x"
  set size 6
  set color red
  hide-turtle
]
create-Signs 1
[
  setxy 38 -33
  set shape "colonies_merged"
  set size 6
  set color brown
  set heading 45
  hide-turtle
]
create-Signs 1
[
  setxy 38 -40
  set shape "queen"
```

```
    set size 8
    set color 33
    set heading 0
    hide-turtle
    ]
  create-Signs 1 ; ***NEW FOR BEEHAVE_BEEMAPP2015***
  [
    setxy 38 -40
    set shape "queenx"
    set size 8
    set color 33
    set heading 0
    hide-turtle
    ]
end

;
**********************************************************************************
**********************************************************************************
************

to Go
  tick
  DailyUpdateProc
  SeasonProc_HoPoMo
  ; Egg laying & development:
  WorkerEggsDevProc
  DroneEggsDevProc
  NewEggsProc
  if Swarming != "No swarming" [ SwarmingProc ]
  WorkerEggLayingProc
  DroneEggLayingProc
  WorkerLarvaeDevProc
  DroneLarvaeDevProc
  NewWorkerLarvaeProc
  NewDroneLarvaeProc
  WorkerPupaeDevProc
  DronePupaeDevProc
  NewWorkerPupaeProc
  NewDronePupaeProc
  WorkerIHbeesDevProc
  DronesDevProc
  BroodCareProc
  NewIHbeesProc
  NewDronesProc
  ; Varroa mite module:
  MiteProc
  ;  if (TotalMites > 0) [ MiteProc ]  ; ***NEW FOR BEEHAVE_BEEMAPP2015***

  BeekeepingProc
  DrawIHcohortsProc
```

```
 ; Foraging module:
 GenericPlotClearProc
 if ( TotalForagers
       + NewForagerSquadronsHealthy * SQUADRON_SIZE
       + NewForagerSquadronsInfectedAsPupae * SQUADRON_SIZE
       + NewForagerSquadronsInfectedAsAdults * SQUADRON_SIZE ) > 0
  [
    Start_IBM_ForagingProc
  ]

 ask turtles
 [
  set label-color black
  ifelse ploidy = 2
  [
    set label number
  ]
  [
    if ploidy = 1
    [
      set label number
    ]
  ]
 ]
 CountingProc
 PollenConsumptionProc
 HoneyConsumptionProc
 DoPlotsProc
end

;
**********************************************************************************
**********************************************************************************
************

to GoTreatmentProc
  ; similar to "Go", but used if colonies don't start on 1st January
  ; (e.g. to mimic empirical colony treatments), called only once by "Setup"
  ; but contains a "repeat"-loop


;;  repeat (INSERT START DAY)
;;  [
;;    Go
;;    set HoneyEnergyStore (MAX_HONEY_ENERGY_STORE / 5)
;;    set PollenStore_g 0.5 * IdealPollenStore_g
;;    ; guarantees survival of colonies before experiment
;;  ]
;;
;;  ask (turtle-set droneEggCohorts droneLarvaeCohorts) [ set number (INSERT NUMBER) ]
;;
```

```
;;  ask (turtle-set dronePupaeCohorts droneCohorts)
;;  [
;;    set number (INSERT NUMBER)
;;    set number_healthy (INSERT NUMBER)
;;    set number_infectedAsPupa (INSERT NUMBER)
;;  ]
;;  ask eggCohorts [ set number (INSERT NUMBER) ]
;;  ask larvaeCohorts [ set number (INSERT NUMBER) ]
;;  ask pupaeCohorts
;;  [
;;    set number (INSERT NUMBER)
;;    set number_Healthy (INSERT NUMBER)
;;    set number_infectedAsPupa (INSERT NUMBER)
;;  ]
;;
;;  ask IHbeeCohorts
;;  [
;;    set number_healthy (INSERT NUMBER)
;;    set number_infectedAsPupa (INSERT NUMBER)
;;    set number_infectedAsAdult (INSERT NUMBER)
;;  ]
;;
;;  set HoneyEnergyStore ENERGY_HONEY_per_g * (INSERT NUMBER OF CELLS WITH HONEY)
;;    ; 1 comb ca. 2*3268 cells (PJK), 1 cell full of honey = 500mg
;;    ; (Schmickl, Crailsheim, HoPoMo)
;;
;;  if Experiment = "INSERT NAME EXPERIMENT A"
;;  [
;;    (INSERT INITIAL CONDITIONS FOR EXERIMENT A)
;;  ]
;;
;;  if Experiment = "INSERT NAME EXPERIMENT B"
;;  [
;;    (INSERT INITIAL CONDITIONS FOR EXERIMENT B)
;;  ]
;;
;;
;;  ask miteOrganisers
;;  [
;;    set droneCellListCondensed n-values (MAX_INVADED_MITES_DRONECELL + 1) [ (INSERT
NUMBER) ]
;;  ]    ; +1 as also the number of mite free cells is stored in this list
;;
;;  StartProc
end

;
**********************************************************************************
**********************************************************************************
************
```

```
to-report FlowerPatchesMaxFoodAvailableTodayREP [ patchID foodType ]
  ; foodType: "Nectar" or "Pollen"
  ; determines the max amount of nectar and pollen available at the patch today
  ; this reporter is ONLY called if ReadInfile = FALSE!!
  ; called by: CreateFlowerPatchesProc (i.e. 1x per run), DailyUpdateProc (i.e. 1x per day),
  ; and       FlowerPatchesUpdateProc (i.e. 1x per foraging round)

  ifelse SeasonalFoodFlow = true
  [
    ; SEASONAL variation of nectar ond pollen availability at RED and
    ; GREEN patch (if SeasonalFoodFlow = ON):
    let patchDayR day + SHIFT_R
    if day + SHIFT_R > 365 [ set patchDayR patchDayR - 365 ]
      ; to shift the seasonal food offer to earlier (+) or later (-) in the year

    let patchDayG day + SHIFT_G
    if day + SHIFT_G > 365 [ set patchDayG patchDayG - 365 ]

    if foodType != "Nectar" and foodType != "Pollen"
    [
      set BugAlarm true
      show "BUG ALARM in FlowerPatchesFoodAvailableTodayREP - Wrong 'foodType' of flower
patch!"
    ]
    if patchID != 0 and patchID != 1
    [
      set BugAlarm true
      show "BUG ALARM in FlowerPatchesFoodAvailableTodayREP - Wrong 'who' of flower patch!"
    ]

    if ReadInfile = true
    [
      set BugAlarm true
      show "BUG ALARM in FlowerPatchesFoodAvailableTodayREP - called although ReadInfile = true!"
    ]

    if patchID = 0 ; "RED" patch
    [
      if foodType = "Nectar"
      [
        report (1 - Season_HoPoMoREP patchDayR []) * QUANTITY_R_l * 1000 * 1000
      ]
      if foodType = "Pollen"
      [
        report (1 - Season_HoPoMoREP patchDayR []) * POLLEN_R_kg * 1000
      ]
    ]

    if patchID = 1 ; "GREEN" patch
    [
      if foodType = "Nectar"
```

```
    [
     report (1 - Season_HoPoMoREP patchDayG []) * QUANTITY_G_l * 1000 * 1000
    ]
    if foodType = "Pollen"
    [
     report (1 - Season_HoPoMoREP patchDayG []) * POLLEN_G_kg * 1000
    ]
   ]
  ]
  [
   ; ELSE (i.e. if SeasonalFoodFlow = FALSE):
   if foodType = "Nectar"
   [
    if patchID = 0 [ report QUANTITY_R_l * 1000 * 1000 ] ; "red" patch
    if patchID = 1 [ report QUANTITY_G_l * 1000 * 1000 ] ; "green" patch
   ]

   if foodType = "Pollen"
   [
    if patchID = 0 [ report POLLEN_R_kg * 1000 ] ; "red" patch
    if patchID = 1 [ report POLLEN_G_kg * 1000 ] ; "green" patch
   ]
  ]
end



;
*******************************************************************************
*******************************************************************************
************

to DailyUpdateProc
 set Day round (ticks mod 365.00001)
 set DeathsAdultWorkers_t 0
 set SumLifeSpanAdultWorkers_t 0
 set DailyMiteFall 0
 set Pupae_W&D_KilledByVirusToday 0
 set NewReleasedMitesToday 0
  ; all (healthy and infected) mites released from cells (mothers+offspring)
  ; on current day (calculated after MiteFall!)

 ask foragerSquadrons [ set km_today 0 ]
 if N_INITIAL_MITES_INFECTED = 0 and AllowReinfestation = false
 [
  if ( count foragerSquadrons with [ infectionState = "infectedAsPupa"]
    + count foragerSquadrons with [ infectionState = "infectedAsAdult"] ) > 0
     or
    ( count IHbeeCohorts with [ number_infectedAsPupa > 0]
    + count IHbeeCohorts with [ number_infectedAsAdult > 0] ) > 0
  [
```

```
    set BugAlarm true
    show "BUG ALARM! Infected bees from out of the blue!"
   ]
  ]

  ask flowerpatches
  [
   ifelse ( quantityMyl < CROPVOLUME * SQUADRON_SIZE
          and
          amountPollen_g < POLLENLOAD * SQUADRON_SIZE )
    [ set shape "fadedFlower" ] ; IF
    [ set shape "Flower" ] ; ELSE = not empty
  ]

  set DailyForagingPeriod Foraging_PeriodREP
  set HoneyEnergyStoreYesterday HoneyEnergyStore
  set PollenStore_g_Yesterday PollenStore_g
  set LostBroodToday 0
  set Queenage Queenage + 1

  ask patch 0 -27 [ set plabel  5] ask patch 0 -32 [ set plabel 10]
  ask patch 0 -37 [ set plabel 15] ask patch 0 -42 [ set plabel 20]
  ask patch 0 -47 [ set plabel 25] ask patch 0 -52 [ set plabel 30]
  ask patch 0 -57 [ set plabel 35] ask patch 1 -58 [ set plabel "age  "]

  set SearchingFlightsToday 0
  set RecruitedFlightsToday 0
  set NectarFlightsToday 0
  set PollenFlightsToday 0
  set EmptyFlightsToday 0
  set DeathsForagingToday 0

  if ReadInfile = false
  [
   ask flowerPatches
   [ ; flower patches are set to the max. amount of nectar and pollen possible today:
    set quantityMyl FlowerPatchesMaxFoodAvailableTodayREP who "Nectar"
    set amountPollen_g FlowerPatchesMaxFoodAvailableTodayREP who "Pollen"
   ]
  ]

 ask flowerPatches
 [
  set nectarVisitsToday 0 set pollenVisitsToday 0
  if detectionProbability < -1
  [
   set BugAlarm true
   user-message "Wrong detection probability! Set 'ModelledInsteadCalcDetectProb' 'false' and re-
start run!"
  ]
 ]
```

```
if ReadInfile = true
[
  set TodaysSinglePatchList []
    ; short list, contains data of current patch and only for today
  set TodaysAllPatchesList []
    ; shorter list, contains data of all patches, but only for today
  let counter (Day - 1)
  repeat N_FLOWERPATCHES
  [
    ; todays data for ALL N_FLOWERPATCHES flower patches are saved in a new,
    ; shorter list (= todaysAllPatchesList)

    set TodaysSinglePatchList (item counter AllDaysAllPatchesList)
      ; this new, shorter list (= todaysAllPatchesList) is comprised of very
      ; short lists (=todaysSinglePatchList) that contain only the data of the
      ; current patch and only for today

    set TodaysAllPatchesList fput TodaysSinglePatchList TodaysAllPatchesList
      ; fput: faster as lput (NetLogo version 4)! however: list is in reversed order!

    set counter counter + 365
    let id item 1 TodaysSinglePatchList ; patch number

    ask flowerpatch id
    [
      set amountPollen_g item 8 TodaysSinglePatchList ; [g]
      if amountPollen_g < 0 [ set amountPollen_g 0 ]
      set quantityMyl (item 10 TodaysSinglePatchList) * 1000 * 1000
        ; [microlitres]  new nectar value from infile (emptied flowers
        ; replenish nectar completely (or are replace by new flowers))

      if quantityMyl < 0 [ set quantityMyl 0 ]
      if id != who [  user-message "Error in id / who!" set BugAlarm true  ]

      if shape != "fadedflower"
      [
        ifelse amountPollen_g > 250
        [ set shape "flowerorange" ]
        [ set shape "flower" ]
      ]
        ; if a "reasonable" amount of pollen available, patch is shown
        ; as 'pollen patch'

      ifelse quantityMyl < CROPVOLUME * SQUADRON_SIZE [ set color grey ]
      [
        set color scale-color red eef 0 50
          ; colour: reddish, dependent on eef, if eff >= 50: white
      ]
    ]
  ] ; ask flowerpatch ID
```

```
  set todaysAllPatchesList reverse todaysAllPatchesList
    ; to correct the reversed order, caused by the fput command
] ; repeat

ask patches [ set pcolor PATCHCOLOR ]

ask hives
[
  set shape "beehiveDeepHive"
    ; # of supers on drawn colony depends on honey store

  if HoneyEnergyStore / ENERGY_HONEY_per_g > 15000 [ set shape "beehive1super" ]
  if HoneyEnergyStore / ENERGY_HONEY_per_g > 30000 [ set shape "beehive2super" ]
  if HoneyEnergyStore / ENERGY_HONEY_per_g > 45000 [ set shape "beehive3super" ]
  if HoneyEnergyStore / ENERGY_HONEY_per_g > 60000 [ set shape "beehive4super" ]
  if HoneyEnergyStore / ENERGY_HONEY_per_g > 75000 [ set shape "beehive5super" ]
  if HoneyEnergyStore / ENERGY_HONEY_per_g > 90000 [ set shape "beehive6super" ]
  if HoneyEnergyStore / ENERGY_HONEY_per_g > 105000 [ set shape "beehive7super" ]
  if HoneyEnergyStore < 0
  [
    if ColonyDied = false
    [
      output-print word "Starvation! Colony died on Day " ticks
    ]
    set ColonyDied true
  ]
] ; ask hives

if (ticks > 1) and (TotalWorkerAndDroneBrood + TotalIHbees + TotalForagers = 0)
[
  if ColonyDied = false
  [
    output-print word "No bees left! Colony died on Day " ticks
  ]
  set ColonyDied true
]

if (Day = 365)
[
  output-type word "31.12.: COLONY SIZE: " (TotalIHbees + TotalForagers)
  output-type "   HONEY STORE [kg]: "
  output-print precision (HoneyEnergyStore / (1000 * ENERGY_HONEY_per_g)) 1
]

if (Day = 365) and (TotalIHbees + TotalForagers < CRITICAL_COLONY_SIZE_WINTER)
[
  if ColonyDied = false
  [
    output-print word "Winter mortality! Colony died on Day " ticks
  ]
```

```
  set ColonyDied true
]

if ColonyDied = true
[
  ask hives [ set color grey ]
    ; grey colony: died! (even if it "recovers" later, it remains grey)

  if stopDead = true
  [
    ask Signs with [shape = "skull"]
    [
      show-turtle
    ]
  ]
  ask patches [ set pcolor black ]
  if stopDead = true
  [
    ask eggCohorts [ set number 0]
    ask larvaeCohorts [ set number 0]
    ask pupaeCohorts
    [
      set number 0
      set number_Healthy 0
      set number_infectedAsPupa 0
    ]
    ask IHbeeCohorts
    [
      set number 0
      set number_Healthy 0
      set number_infectedAsPupa 0
      set number_infectedAsAdult 0
    ]
    ask foragerSquadrons [ die ]
    ask droneEggCohorts [ set number 0]
    ask droneLarvaeCohorts [ set number 0]
    ask dronePupaeCohorts
    [
      set number 0
      set number_Healthy 0
      set number_infectedAsPupa 0
    ]
    ask droneCohorts [ set number 0 ]
  ] [
]

    set number 0
    set number_Healthy 0          ; ***NEW FOR BEEHAVE_BEEMAPP2015***
    set number_infectedAsPupa 0   ; ***NEW FOR BEEHAVE_BEEMAPP2015***
  ]
]
```

```
 ]

 if ReadBeeMappFile = true [ BeeMappCorrectionProc ]   ;  ***NEW FOR
BEEHAVE_BEEMAPP2015***
end

;
**********************************************************************************
**********************************************************************************
************

to-report Season_HoPoMoREP [ today parameterList ]
  ; see Schmickl&Crailsheim2007: p.221 and p.230
  ; Values HoPoMo: x1 385; x2 30; x3 36; x4 155; x5 30

  let x1 385  ;385
  let x2 25 ; (earlier increase in egg laying rate than in HoPoMo)
  let x3 36    ; 36
  let x4 155   ;155  ; Day of max. egg laying
  let x5 30    ;30
  if empty? parameterList = false
  [
    set x1 item 0 parameterList
    set x2 item 1 parameterList
    set x3 item 2 parameterList
    set x4 item 3 parameterList
    set x5 item 4 parameterList
  ]
  let seas1 (1 - (1 / (1 + x1 * e ^ (-2 * today / x2))))
  let seas2 (1 / (1 + x3 * e ^ (-2 * (today - x4) / x5)))
  ifelse seas1 > seas2
    [ report seas1 ]
    [ report seas2 ]
end

;
**********************************************************************************
**********************************************************************************
************

to SeasonProc_HoPoMo
  ; see Schmickl&Crailsheim2007: p.221 and p.230

  set HoPoMo_seasont Season_HoPoMoREP day []
    ; calls to-report SeasonProc_HoPoMoREP to calculate the HoPoMo seasonal
    ; factor on basis of "day" and of a parameter list ("[]"), which is empty in
    ; this case but could contain 5 values: x1..x5
end

;
**********************************************************************************
```

```
**********************************************************************************
************

to NewEggsProc
 ; CALLED BY WorkerEggLayingProc   see: HoPoMo p.222 & p.230, ignoring ELRstoch
 let ELRt_HoPoMo (MAX_EGG_LAYING * (1 - HoPoMo_seasont))
 if EMERGING_AGE <= 0 [ set BugAlarm true show "EMERGING_AGE <= 0" ]
 let ELRt_IH (TotalIHbees
    + TotalForagers * FORAGER_NURSING_CONTRIBUTION)
    * MAX_BROOD_NURSE_RATIO / EMERGING_AGE
     ; EMERGING_AGE = 21: total developmental time of worker brood

 let ELRt ELRt_HoPoMo
  ; egg laying rate follows a seasonal pattern as described in
  ; HoPoMo (Schmickl & Crailsheim 2007)

 if EggLaying_IH = true and ELRt_IH < ELRt_HoPoMo
  ; if EggLaying_IH SWITCH is on and not enough nurse bees are available,
  ; the egg laying rate is reduced to ELRt_IH
 [
  set ELRt ELRt_IH
 ]

 if ELRt > MAX_EGG_LAYING
 [
  set ELRt MAX_EGG_LAYING
 ]

 ;   LIMITED BROOD NEST:
 if TotalWorkerAndDroneBrood + ELRt > MAX_BROODCELLS
 [
  set ELRt MAX_BROODCELLS - TotalWorkerAndDroneBrood
 ]

 set NewWorkerEggs round ELRt  ; ROUND! in contrast to HoPoMo

 ; CALCULATION OF DRONE EGGS:
 set NewDroneEggs floor(NewWorkerEggs * DRONE_EGGS_PROPORTION)
 if Day >= SEASON_STOP
   - ( DRONE_HATCHING_AGE
   -   DRONE_PUPATION_AGE
   -   DRONE_EMERGING_AGE )
 [
  set NewDroneEggs 0
 ] ; no more drone brood at end of season (however: Season set to day 1 - 365)

 ; AGEING OF QUEEN - based on deGrandi-Hofmann, BEEPOP:
 if QueenAgeing = true ; GUI: "switch"
 [
  let potentialEggs (MAX_EGG_LAYING
     + (-0.0027 * Queenage ^ 2)
```

```
      + (0.395 * Queenage))
        ; Beepops potential egglaying Pt
   set NewWorkerEggs round (NewWorkerEggs * (potentialEggs / MAX_EGG_LAYING) )
  ]

  ; no egg-laying of young queen (also if QUEEN_AGEING = false!):
  ask signs with [ shape = "queenx" ] [ hide-turtle ]  ; ***NEW FOR BEEHAVE_BEEMAPP2015***
  if Queenage <= 10
  [
   set NewWorkerEggs 0
     ; Winston p. 203: 5-6d until sexually mature, 2-4d for orientation and mating flight, mating
     ; can be postponed for 4 weeks if weather is bad

   set NewDroneEggs 0
   ask signs with [ shape = "queenx" ] [ show-turtle ]  ; ***NEW FOR BEEHAVE_BEEMAPP2015***
  ]
  if NewWorkerEggs < 0 [ set NewWorkerEggs 0 ]
  if NewDroneEggs < 0 [ set NewDroneEggs 0 ]
end

;
*******************************************************************************
*******************************************************************************
************

to SwarmingProc

  ; # total brood triggers swarming
  ; PRE_SWARMING_PERIOD: 3d of preparation before swarming
  ; SwarmingDate: set to 0 in Param.Proc and in SwarmingProc (after swarming and on day 365)

  let fractionSwarm 0.6  ; 0.6 ; Winston p. 187
  let broodSwarmingTH 17000 ; Fefferman & Starks 2006 (model)
  let lastSwarmingDate 199; Winston 1980: prime: 14.05.(134) after swarm: 18.07.(199)
   ; McLellan, Rowland 1986: 162 (modelled),
  if TotalWorkerAndDroneBrood > broodSwarmingTH and SwarmingDate = 0 and day <=
(lastSwarmingDate - PRE_SWARMING_PERIOD)
  [
   set SwarmingDate (day + PRE_SWARMING_PERIOD)
  ]

  if day = SwarmingDate
    and Swarming = "Swarm control"
  [
   output-type "Swarming (prevented) on day: " output-print day
  ]

  if day >= SwarmingDate - PRE_SWARMING_PERIOD
    and day <= SwarmingDate
  [
   if Swarming = "Swarming (parental colony)"
```

```
[ ; Swarm PREPARATION of PARENTAL colony:
  set NewDroneEggs 0
  set NewWorkerEggs 0
  if  day = SwarmingDate
  [ ; SWARMING of PARENTAL colony:
    set Queenage -7
      ; a new queen is left in the hive, still in a capped cell, ca. 7d
      ; before she emerges (Winston p. 187)

    ; Winston p. 185: 36mg honey is taken by a swarming bee:
    set HoneyEnergyStore HoneyEnergyStore
       - (( TotalForagers + TotalIHbees) * 0.036 * ENERGY_HONEY_per_g)
       * fractionSwarm

    ; (1-fractionSwarm) of all healthy & infected in-hive bees stay in the hive:
    ask IHbeeCohorts
    [
      set number_Healthy round (number_Healthy * (1 - fractionSwarm))
      set number_infectedAsPupa round (number_infectedAsPupa * (1 - fractionSwarm))
      set number_infectedAsAdult round (number_infectedAsAdult * (1 - fractionSwarm))
      set number number_Healthy + number_infectedAsPupa + number_infectedAsAdult
    ]

    ; (1-fractionSwarm) of all healthy & infected drones stay in the hive:
    ask droneCohorts
    [
      set number_Healthy round (number_Healthy * (1 - fractionSwarm))
      set number_infectedAsPupa round (number_infectedAsPupa * (1 - fractionSwarm))
      set number number_Healthy + number_infectedAsPupa
    ]

    ; fractionSwarm foragers leave the colony and are considered to be dead in the model:
    ask foragerSquadrons
    [
      if random-float 1 < fractionSwarm [ die ]
    ] ; LEAVING foragers are treated as being dead

    ; the phoretic mite population in the hive is reduced:
    set PhoreticMites round (PhoreticMites * (1 - fractionSwarm))
    output-type "Swarming on day: " output-print day
    set SwarmingDate 0  ; allows production of after swarms
  ]
]


if Swarming = "Swarming (prime swarm)"
[ ; Swarm PREPARATION of PRIME SWARM:
  set NewDroneEggs 0
  set NewWorkerEggs 0
  if  day = SwarmingDate
  [ ; Swarming of PRIME SWARM:
```

```
    ask (turtle-set eggCohorts larvaeCohorts droneEggCohorts droneLarvaeCohorts)
    [ ; all brod is left behind and hence removed from the smulation:
      set number 0
    ]
    ask (turtle-set pupaeCohorts dronePupaeCohorts)
    [
      set number 0
      set number_infectedAsPupa 0
      set number_healthy 0
    ]
    set NewWorkerLarvae 0
    set NewDroneLarvae 0
    set NewWorkerPupae 0
    set NewDronePupae 0
    ask IHbeeCohorts
    [ ; fractionSwarm of all healthy & infected in-hive bees join the swarm
      set number_Healthy round (number_Healthy * fractionSwarm)
      set number_infectedAsPupa round (number_infectedAsPupa * fractionSwarm)
      set number_infectedAsAdult round (number_infectedAsAdult * fractionSwarm)
      set number number_Healthy + number_infectedAsPupa + number_infectedAsAdult
    ]

    ask droneCohorts
    [ ; fractionSwarm of all healthy & infected drones join the swarm
      set number_Healthy round (number_Healthy * fractionSwarm)
      set number_infectedAsPupa round (number_infectedAsPupa * fractionSwarm)
      set number number_Healthy + number_infectedAsPupa
    ]

    ask foragerSquadrons
    [ ; (1 - fractionSwarm) foragers do not join the swarm and hence die (in the model):
      if random-float 1 < (1 - fractionSwarm) [ die ]
    ]

    ask miteOrganisers [ die ]
      ; mites in brood cells are left behind in the old colony

    ; the phoretic mite population in the swarm is reduced:
    set PhoreticMites round (PhoreticMites * fractionSwarm)
    set PollenStore_g 0
    set HoneyEnergyStore
        ((TotalForagers + TotalIHbees)
          * 36 * ENERGY_HONEY_per_g) / 1000
      ; Winston p. 185: 36mg honey per bee during swarming
    output-type "Swarming on day: "
    output-print day
    set SwarmingDate 0  ; allows production of after swarms
   ] ; if  day = SwarmingDate ..
  ] ; if Swarming = "Swarming (prime swarm)"   ,,
] ; if SwarmingDate > 0 and ..
```

```
  if Swarming = "Swarm (daughter colony)"
    and day > SwarmingDate
    and day <= SwarmingDate + POST_SWARMING_PERIOD   ; DAUGHTER COLONY AFTER SWARMING
(0d period)
  [ ; no eggs can be laid, no food stored, as long as they have no new home..
   set NewDroneEggs 0
   set NewWorkerEggs 0
   set PollenStore_g 0
   set Aff MAX_AFF
   if HoneyEnergyStore >
    (((TotalForagers + TotalIHbees) * CROPVOLUME) / 1000)
      * 1.36 * ENERGY_HONEY_per_g
   [
    set HoneyEnergyStore (((TotalForagers + TotalIHbees) *
     CROPVOLUME) / 1000) * 1.36 * ENERGY_HONEY_per_g
   ]
  ]
  ; resetting SwarmingDate to zero at the end of a year:
  if day = 365 [ set SwarmingDate 0 ]
end

;
**********************************************************************************
**********************************************************************************
************

to WorkerEggLayingProc  ; creation of worker eggs
  create-eggCohorts 1 ;
  [
   set shape "circle"
   set number NewWorkerEggs
   set age 0
   setxy 3 0
   set color blue
   set ploidy 2
  ]
end

;
**********************************************************************************
**********************************************************************************
************

to DroneEggLayingProc  ; creation of drone eggs
  create-DroneEggCohorts 1 ;
  [
   set shape "circle"
   set number NewDroneEggs
   if Day < DRONE_EGGLAYING_START or Day > DRONE_EGGLAYING_STOP [ set number 0 ]
   set age 0
   setxy -5 0
```

```
   set color blue
   set ploidy 1
  ]
end

;
*******************************************************************************
*******************************************************************************
************

to WorkerEggsDevProc ; ageing, deletion of oldest cohort
  ask eggCohorts
  [
   set age age + 1
   fd 1   ; turtle moves one step (display)
   set number (number - random-poisson (number * MORTALITY_EGGS))
   if number < 0 [ set number 0 ]
    ; random mortality, based on Poisson distribution

   if age = HATCHING_AGE [ set NewWorkerLarvae number ]
   if age >= HATCHING_AGE [ die ]
  ]
end

;
*******************************************************************************
*******************************************************************************
************

to DroneEggsDevProc  ; ageing, deletion of oldest cohort
  ask droneEggCohorts
  [
   set age age + 1
   set number (number - random-poisson (number * MORTALITY_DRONE_EGGS))
   if number < 0 [ set number 0 ]  ; random mortality, based on Poisson distribution
   if age = DRONE_HATCHING_AGE [ set NewDroneLarvae number ]
   if age >= DRONE_HATCHING_AGE [ die ]
   fd 1    ; turtle moves one step (display)
  ]
end

;
*******************************************************************************
*******************************************************************************
************

to NewWorkerLarvaeProc ; creation of worker larvae
  create-larvaeCohorts 1
  [
    set number NewWorkerLarvae   ; the cohort size
    set age HATCHING_AGE
```

```
    set shape "circle"   ; shape
    set color yellow
    setxy 3 (- age)
    set ploidy 2     ; worker larvae are diploid
  ]
end

;
*******************************************************************************
*******************************************************************************
************

to NewDroneLarvaeProc  ; creation of drone larvae
  create-droneLarvaeCohorts 1
  [
    set shape "circle"
    set number NewDroneLarvae   ; the cohort size
    set age DRONE_HATCHING_AGE
    set color yellow
    setxy -5 (- age)
    set ploidy 1    ; drone larvae are haploid
  ]
end

;
*******************************************************************************
*******************************************************************************
************

to WorkerLarvaeDevProc ; ageing of cohort
  ask larvaeCohorts
  [
    set age age + 1
    fd 1    ; turtle moves one step (display)
    set numberDied 0
    set numberDied random-poisson (number * MORTALITY_LARVAE)
    if numberDied > number [ set numberDied number ]
      ; random mortality, based on Poisson distribution

    set number number - numberDied
    if (numberDied > 0)
      and ( age > INVADING_WORKER_CELLS_AGE )
      and (TotalMites > 0)
    [
      MitesReleaseProc invadedByMiteOrganiserID ploidy numberDied "dyingBrood"
    ]

    if age = PUPATION_AGE
    [
      set NewWorkerPupae number
      set SaveWhoWorkerLarvaeToPupae who   ; "Who" is stored as a global variable
```

```
    set SaveInvadedMOWorkerLarvaeToPupae invadedByMiteOrganiserID
   ]
   if age >= PUPATION_AGE [ die ]
  ]
end

;
**********************************************************************************
**********************************************************************************
************

to DroneLarvaeDevProc ; ageing of cohort
  ask droneLarvaeCohorts
  [
   set age age + 1
   set numberDied 0
   set numberDied random-poisson (number * MORTALITY_DRONE_LARVAE)
   if numberDied > number [ set numberDied number ]
    ; random mortality, based on Poisson distribution
   set number number - numberDied

   if (numberDied > 0)
      and (-age > INVADING_DRONE_CELLS_AGE-)
      and (TotalMites > 0)
   [
    MitesReleaseProc invadedByMiteOrganiserID ploidy numberDied "dyingBrood"
   ] ; variables correspond to [ miteOrganiserID ploidyMO diedBrood ]

   fd 1
   if age = DRONE_PUPATION_AGE
   [
    set NewDronePupae number
    set SaveWhoDroneLarvaeToPupae who   ; "Who" is stored as a global variable
    set saveInvadedMODRONELarvaeToPupae invadedByMiteOrganiserID
   ]
   if age >= DRONE_PUPATION_AGE [ die ]
  ]
end

;
**********************************************************************************
**********************************************************************************
************

to NewWorkerPupaeProc
  create-pupaeCohorts 1
  [
   set shape "circle"   ; shape of the turtle as shown in the GUI
   set number NewWorkerPupae  ; cohort size
   set number_healthy number ; all newly created pupae are healthy
   set age PUPATION_AGE ; age of the cohort
```

```
   setxy 3 (- age)   ; xy position of the turtle in the Netlogo world
   set color brown    ; color of the turtle
   set ploidy 2    ; worker pupae are diploid
   set invadedByMiteOrganiserID SaveInvadedMOWorkerLarvaeToPupae
    ; saves "invadedByMiteOrganiserID" of the old larvaeCohort that has now developed
    ; into a pupaeCohort
   let saveWho who
    ; saves "who" for the following command (transition of larvae to pupae results in the
    ; death of larvae turtles, hence: ensuing pupae turtles have a different "who")
   ask miteOrganisers with [ invadedWorkerCohortID = SaveWhoWorkerLarvaeToPupae ]
   [
    set invadedWorkerCohortID saveWho
   ] ; miteOrganiser updates its value for the invadedWorkerCohortID
 ]
end

;
**********************************************************************************
**********************************************************************************
************
to NewDronePupaeProc
  create-dronePupaeCohorts 1
  [
   set shape "circle"
   set number NewDronePupae
   set number_healthy number ; all newly created pupae are healthy
   set age DRONE_PUPATION_AGE
   setxy -5 (- age)
   set color brown
   set ploidy 1
   set invadedByMiteOrganiserID SaveInvadedMODroneLarvaeToPupae
    ; saves "invadedByMiteOrganiserID" of the old larvaeCohort that has
    ; now developed into a pupaeCohort

   let saveWho who
    ; saves "who" for the next line (transition of larvae to pupae results
    ; in the death of larvae turtles, hence: ensuing pupae turtles
    ; have a different "who")

   ask miteOrganisers with [ invadedDroneCohortID = SaveWhoDroneLarvaeToPupae ]
   [
    set invadedDroneCohortID saveWho
   ] ; miteOrganiser updates its value for the invadedDroneCohortID
  ]
end

;
**********************************************************************************
**********************************************************************************
************
```

```
to WorkerPupaeDevProc
 ; ageing of cohort, oldest cohort may emerge and release mites
 ask pupaeCohorts
 [
  set age age + 1
  fd 1
  set numberDied 0
  set numberDied random-poisson (number * MORTALITY_PUPAE)
  if numberDied > number [ set numberDied number ]
   ; random mortality, based on Poisson distribution
  set number number - numberDied
  set number_healthy number_healthy - numberDied
   ; all pupae are healthy as infection takes place (in the model)
   ; at emergence - and if not..

  if number_infectedAsPupa > 0
  [
   set BugAlarm true
   show "BUG ALARM!!! number_infectedAsPupa > 0 in WorkerPupaeDevProcs!"
  ] ;  .. raise a bug alarm!

  if (numberDied > 0) and (TotalMites > 0)
  [
   MitesReleaseProc invadedByMiteOrganiserID ploidy numberDied "dyingBrood"
  ] ; variables correspond to [ miteOrganiserID ploidyMO diedBrood ]

  if age = EMERGING_AGE
  [
   if (number > 0) and (TotalMites > 0)
   [
    MitesReleaseProc invadedByMiteOrganiserID 2 0 "emergingBrood"
   ]  ; invadedByMiteOrganiserID ploidy = 2 numberDied = 0

   set NewIHbees number
   set NewIHbees_healthy number_healthy
  ]

  if age >= EMERGING_AGE [ die ]
 ]
end

;
**********************************************************************************
**********************************************************************************
************

to DronePupaeDevProc
 ; ageing of cohort, oldest cohort may emerge and release mites
 ask dronePupaeCohorts
 [
  set age age + 1
```

```
    fd 1  ; turtle moves one step (display)
    set numberDied 0
    set numberDied random-poisson (number * MORTALITY_DRONE_PUPAE)
    if numberDied > number [ set numberDied number ]
    set number number - numberDied
    set number_healthy number_healthy - numberDied
      ; all pupae are healthy as infection takes place (in the model) at
      ; emergence - and if not..

    if number_infectedAsPupa > 0
    [
      set BugAlarm true
      show "BUG ALARM!!! number_infectedAsPupa > 0 in DronePupaeDevProcs!"
    ] ;  .. raise a bug alarm!
    if (numberDied > 0) and (TotalMites > 0)
    [
      MitesReleaseProc invadedByMiteOrganiserID ploidy numberDied "dyingBrood"
    ] ; variables correspond to [ miteOrganiserID ploidyMO diedBrood ]
    if age = DRONE_EMERGING_AGE
    [
      if (number > 0) and (TotalMites > 0)
      [
        MitesReleaseProc invadedByMiteOrganiserID 1 0 "emergingBrood"
      ]   ; invadedByMiteOrganiserID ploidy = 1 numberDied = 0
      set NewDrones number
      set NewDrones_healthy number_healthy ]
    if age >= DRONE_EMERGING_AGE [ die ]
  ]
end

;
*******************************************************************************
*******************************************************************************
************

to NewIHbeesProc
  create-IHbeeCohorts 1
  [
    set shape "circle"
    set number NewIHbees ; all new IH bees
    set number_healthy NewIHbees_healthy ; new, healthy IH bees
    set number_infectedAsPupa number - number_healthy
      ; the others were infected during pupal phase

    set number_infectedAsAdult 0
      ; adult workers hadn't had any chance to become infected so far..

    set age 0
    set color orange
    setxy 3 (- age - EMERGING_AGE - 1)
    set ploidy 2
```

```
  ]
end

;
**********************************************************************************
**********************************************************************************
************

to NewDronesProc
  create-DroneCohorts 1
  [
    set shape "circle"
    set number NewDrones ; all new drones
    set number_healthy NewDrones_healthy ; new, healthy drones
    set number_infectedAsPupa number - number_healthy ; the others are infected
    set age 0
    set color grey
    setxy -5 (- age - DRONE_EMERGING_AGE - 1)
    set ploidy 1
  ]
end

;
**********************************************************************************
**********************************************************************************
************

to AffProc
  ; calculates the actual age of first foraging on basis of nectar stores and
  ; brood/nurse ratio - called by WorkerIHbeesDevProc

  let affYesterDay Aff  ; the current (= yesterday's) Aff is saved
  let pollenTH  0.5
  let proteinTH 1
  let honeyTH 35 * (DailyHoneyConsumption / 1000) * ENERGY_HONEY_per_g
    ; min. desired honey store lasts for 35 days (arbitrarily chosen)
  let broodTH 0.1
  let foragerToWorkerTH 0.3  ; like in Beshers et al. 2001

  ; POLLEN criterion:
  if PollenStore_g / IdealPollenStore_g < pollenTH [ set Aff Aff - 1 ]

  ; PROTEIN criterion:
  if proteinFactorNurses < proteinTH [ set Aff Aff - 1 ]

  ; HONEY criterion:
  if HoneyEnergyStore < honeyTH [ set Aff Aff - 2 ]

  ; FORAGER TO WORKER criterion:
  if (TotalIHbees > 0)
    and (TotalForagers / TotalIHbees < foragerToWorkerTH)
```

```
 [
  set Aff Aff - 1
 ]


 ; BROOD TO NURSES criterion:
 if ((TotalIHbees
    + TotalForagers * FORAGER_NURSING_CONTRIBUTION) * MAX_BROOD_NURSE_RATIO)
    > 0
  and
    TotalWorkerAndDroneBrood / ((TotalIHbees
    + TotalForagers * FORAGER_NURSING_CONTRIBUTION) * MAX_BROOD_NURSE_RATIO)
    > broodTH
 [
  set Aff Aff + 2
 ]

 ; to reduce strong deviations from the base Aff:
 if affYesterDay < AFF_BASE - 7 [ set Aff Aff + 1 ]
 if affYesterDay > AFF_BASE + 7 [ set Aff Aff - 1 ]

 ; Aff can be changed only by +-1 per day:
 if Aff < affYesterDay [ set Aff affYesterDay - 1 ]
 if Aff > affYesterDay [ set Aff affYesterDay + 1 ]

 ; MIN and MAX values for Aff:
 if Aff < MIN_AFF [ set Aff MIN_AFF ]
 if Aff > MAX_AFF [ set Aff MAX_AFF ]

end

;
**********************************************************************************
**********************************************************************************
************

to WorkerIHbeesDevProc
 ; ageing of IH bees, mortality for healthy and infected IH-workers,
 ; calls CalculateAffProc, calculation of # new foragerSquadrons

 let overagedIHbees 0
  ; bees with age > Aff but have to remain in the last IH cohort, as number < SQUADRON_SIZE

 AffProc
  ; in the AffProc today's age of first foraging (Aff) is calculated
 foreach reverse sort IHbeeCohorts
  ; cohorts have to be asked in order of their age (i.e. in reverse order of
  ; their "who") otherwise over-aged bees vanish with a 50% chance
 [
  ask ?
  [
```

```
let deathsCounter 0
  ; # of bees dying in this cohort at current time step

set age age + 1
fd 1 ; turtle moves one step (display)

; MORTALITY
; healthy bees:
set deathsCounter random-poisson (number_healthy * MORTALITY_INHIVE)
if deathsCounter > number_healthy [ set deathsCounter number_healthy ]
  ; random mortality, based on Poisson distribution

set number_healthy number_healthy - deathsCounter
  ; deathCounter: dead HEALTHY bees

; infectedAsPupa:
set deathsCounter
  random-poisson (number_infectedAsPupa * MORTALITY_INHIVE_INFECTED_AS_PUPA)
if deathsCounter > number_infectedAsPupa
[
  set deathsCounter number_infectedAsPupa
]  ; random mortality, based on Poisson distribution

set number_infectedAsPupa number_infectedAsPupa - deathsCounter
  ; deathCounter now: dead INFECTED bees

; infectedAsAdults:
set deathsCounter
  random-poisson (number_infectedAsAdult * MORTALITY_INHIVE_INFECTED_AS_ADULT)
if deathsCounter > number_infectedAsAdult
[
  set deathsCounter number_infectedAsAdult
]  ; random mortality, based on Poisson distribution
set number_infectedAsAdult number_infectedAsAdult - deathsCounter
  ; deathCounter now: dead INFECTED bees

set deathsCounter number - number_healthy
  - number_infectedAsPupa - number_infectedAsAdult
  ; deathCounter is now set to the TOTAL number of dead bees

set number number - deathsCounter
  ; # of bees in this cohort is reduced by # of dead bees

set DeathsAdultWorkers_t DeathsAdultWorkers_t
  + deathsCounter
  ; sums up # of adult workers dying in current timestep to calculate
  ; mean lifespan of adult bees

set SumLifeSpanAdultWorkers_t SumLifeSpanAdultWorkers_t
  + (deathsCounter * age)
  ; sums up lifespan of adult workers dying in current timestep
```

```
set InhivebeesDiedToday DeathsAdultWorkers_t

; ONSET OF FORAGING
if age >= Aff
[
 ; new healthy foragerSquadrons:
 set NewForagerSquadronsHealthy
  floor (number_healthy / SQUADRON_SIZE) + NewForagerSquadronsHealthy
 set overagedIHbees number_healthy mod SQUADRON_SIZE
 ask IHbeeCohorts with [ age = Aff - 1 ]
 [
  set number number + overagedIHbees
  set number_healthy number_healthy + overagedIHbees
 ]
  ; overaged bees would vanish here without "reverse sort", as there
  ; might be no IHbeeCohort with age = Aff - 1! (50% chance)

 ; new foragerSquadrons, which were infected as pupae:
 set NewForagerSquadronsInfectedAsPupae
  floor (number_infectedAsPupa / SQUADRON_SIZE)
      + NewForagerSquadronsInfectedAsPupae

 set overagedIHbees number_infectedAsPupa mod SQUADRON_SIZE
 ask IHbeeCohorts with [ age = Aff - 1 ]
 [
  set number number + overagedIHbees
  set number_infectedAsPupa number_infectedAsPupa + overagedIHbees
 ]
  ; overaged bees would vanish here without "reverse sort", as there might
  ; be no IHbeeCohort with age = Aff - 1! (50% chance)

 ; new infectedAsAdults foragerSquadrons:
 set NewForagerSquadronsInfectedAsAdults
  floor (number_infectedAsAdult / SQUADRON_SIZE)
      + NewForagerSquadronsInfectedAsAdults

 set overagedIHbees number_infectedAsAdult mod SQUADRON_SIZE
 ask IHbeeCohorts with [ age = Aff - 1 ]
 [
  set number number + overagedIHbees
  set number_infectedAsAdult number_infectedAsAdult + overagedIHbees
 ]
  ; overaged bees would vanish here without "reverse sort", as there might
  ; be no IHbeeCohort with age = Aff - 1! (50% chance)
        ]
if age >= Aff
[
 set plabel ""
 die
]
```

```
   ] ; ask ?
  ] ; foreach reverse sort IHbeeCohorts
end

;
**********************************************************************************
**********************************************************************************
************

to DronesDevProc
  ; ageing of cohort, mortality for healthy and infected drones
  ask DroneCohorts [
    fd 1
    set age age + 1

    ; MORTALITY:
    set number_healthy (number_healthy -
       random-poisson (number_healthy * MORTALITY_DRONES))
    if number_healthy < 0 [ set number_healthy 0 ]

    set number_infectedAsPupa
     ( number_infectedAsPupa
       - random-poisson (number_infectedAsPupa * MORTALITY_DRONES_INFECTED_AS_PUPAE) )

    if number_infectedAsPupa < 0 [ set number_infectedAsPupa 0 ]
    set number number_healthy + number_infectedAsPupa
     ; total number of drones = healthy + infected drones
    if age >= DRONE_LIFESPAN [ die ]
  ]
end

;
**********************************************************************************
**********************************************************************************
************

to BroodCareProc
  ; checks if enough nurses are present and, if not, kills excess of drone and
  ; worker brood; order of dying: 1. droneEggCohorts 2. droneLarvaeCohorts
  ; 3. eggCohorts 4. larvaeCohorts 5. dronePupaeCohorts 6. pupaeCohorts

  let lackNurses false
    ; all kind of brood might die due to lack of nurse bees..
  let lackProtein false
    ; .. or (drone&worker) LARVAE may die due to lack of protein in brood food

  if ticks > 1 [ CountingProc ]
    ; current # of IH-bees and brood, cannot be called in time step 1, as
    ; counting foragerSquadrons results wrongly in 0

  set ExcessBrood
```

```
  ceiling ( TotalWorkerAndDroneBrood
  - (TotalIHbees + TotalForagers * FORAGER_NURSING_CONTRIBUTION)
  *  MAX_BROOD_NURSE_RATIO )
    ; rounded up! totalWorkerDroneBrood: all brood stages of drones & workers;
    ; Nursing: also foragers are assumed to contribute (partly) to brood care

ifelse ExcessBrood > 0
[
  set lackNurses true
  ask signs with [shape = "beelarva_x2"]
  [
    show-turtle
    set label ExcessBrood
  ]
]
[
  ask signs with [shape = "beelarva_x2"]
  [
    hide-turtle
  ]
]

let starvedBrood ceiling ((TotalDroneLarvae + TotalLarvae) * (1 - ProteinFactorNurses))
  ; larvae require protein and may die if jelly contains not enough proteins

if starvedBrood > 0 [ set lackProtein true ]
if starvedBrood > ExcessBrood [ set ExcessBrood starvedBrood ]
  ; excess of brood is either determined by lack of nurses or lack of protein

set LostBroodToday LostBroodToday + ExcessBrood
set LostBroodTotal LostBroodTotal + ExcessBrood
let stillToKill ExcessBrood
  ; keeps track of the amount of brood that is still to be killed

if ExcessBrood > 0
[ ; whenever a brood cell dies, the corresponding miteOrganiser is updated in the
  ; releaseMitesProc! (only for pupae and oldest larvae as eggs and young larvae are
  ; not invaded by mites

  if lackNurses = true
  [
    foreach reverse sort DroneEggCohorts
    [
      ask ?    ; young drone eggs die first if not enough nurses are available
      [ while [ (stillToKill * number) > 0 ]
        [
          set number number - 1
          set stillToKill stillToKill - 1
        ]
      ]
    ]
```

```
]

if lackNurses = true or lackProtein = true
[
 foreach reverse sort DroneLarvaeCohorts
 [
  ask ?
  [
   while [ (stillToKill * number) > 0 ]
   [ set number number - 1 set stillToKill stillToKill - 1
    if age > INVADING_DRONE_CELLS_AGE and (TotalMites > 0)
    [
     MitesReleaseProc invadedByMiteOrganiserID ploidy 1 "dyingBrood"
    ]
     ; Died brood: always 1! calls releaseMitesProc and transfers variables
     ; (correspond to [ miteOrganiserID ploidyMO diedBrood ])
   ]
  ]
 ]
] ; if lackNurses = true or lackProtein = true

if lackNurses = true
[
 foreach reverse sort EggCohorts
 [
  ask ?
  [
   while [ (stillToKill * number) > 0 ]
   [
    set number number - 1
    set stillToKill stillToKill - 1
   ]
  ]
 ]
] ;if lackNurses = true
; (stillToKill * number): BOTH, number AND stillToKill have to be > 0 to continue "while"

if lackNurses = true or lackProtein = true
[
 foreach reverse sort larvaeCohorts
 [
  ask ?
  [
   while [ (stillToKill * number) > 0 ]
   [
    set number number - 1 set stillToKill stillToKill - 1
    if age > INVADING_WORKER_CELLS_AGE and (TotalMites > 0)
    [
     MitesReleaseProc invadedByMiteOrganiserID ploidy 1 "dyingBrood"
    ]
     ; calls releaseMitesProc and transfers variables (correspond
```

```
        ; to [ miteOrganiserID ploidyMO diedBrood ])
      ]
    ]
  ]
] ; if lackNurses = true or lackProtein = true

if lackNurses = true
[
  foreach reverse sort DronePupaeCohorts
  [
    ask ?
    [
      while [ (stillToKill * number) > 0 ]
      [
        ifelse random number <= number_healthy ; choose a random pupal cell
          [ set number_healthy number_healthy - 1 set number number - 1  ]
            ; IF pupa is healthy, then number_healthy and (total) number are decreased by one
          [ set number_infectedAsPupa number_infectedAsPupa - 1 set number number - 1 ]
            ; ELSE number_infectedAsPupa and (total) number are decreased by one
        set stillToKill stillToKill - 1
        if (TotalMites > 0)
        [
          MitesReleaseProc invadedByMiteOrganiserID ploidy 1 "dyingBrood"
        ]
      ]
    ]
  ]
] ;   if lackNurses = true

if lackNurses = true
[
  foreach reverse sort pupaeCohorts
  [
    ask ?
    [
      while [ (stillToKill * number) > 0 ]
      [
        ifelse random number <= number_healthy  ; choose a random pupal cell
          [ set number_healthy number_healthy - 1 set number number - 1  ]
            ; IF pupa is healthy, then number_healthy and (total) number are decreased by one
          [ set number_infectedAsPupa number_infectedAsPupa - 1 set number number - 1 ]
            ; ELSE number_infectedAsPupa and (total) number are decreased by one
        set stillToKill stillToKill - 1
        if (TotalMites > 0)
        [
          MitesReleaseProc invadedByMiteOrganiserID ploidy 1 "dyingBrood"
        ]
      ]
    ]
  ]
] ; if lackNurses = true
```

```
   if stillToKill > 0
   [
    set BugAlarm true
    output-show (word ticks " BUG ALARM! stillToKill > 0")
   ]
  ] ; end IF ExcessBrood > 0

end

;
**********************************************************************************
**********************************************************************************
************

to DrawIHcohortsProc
  ; # bees in IH cohorts (workers & drones, brood & adults) are drawn as coloured bars

  ask (turtle-set eggCohorts larvaeCohorts pupaeCohorts)
  [                                          ;   WORKERS
    set heading 90
    fd 1
    repeat ceiling( 10 * number / STEPWIDTH)
    [
      fd 0.1
      set pcolor color
    ]
    set heading 180 setxy 3 (- age)
  ]

  ask IHbeeCohorts
  [
    set heading 90
    fd 1
    repeat ceiling( 10 * number_healthy / STEPWIDTH)
    [
      fd 0.1
      set pcolor color
    ]
    repeat ceiling( 10 * number_infectedAsAdult / STEPWIDTH)
    [
      fd 0.1
      set pcolor (color - 1)
    ]
    repeat ceiling( 10 * number_infectedAsPupa / STEPWIDTH)
    [
      fd 0.1
      set pcolor (color - 2)
      ]
    set heading 180
    setxy 3 (- age - EMERGING_AGE - 1)
```

```
 ] ; ask IHbeeCohorts

 ask (turtle-set droneEggCohorts droneLarvaeCohorts dronePupaeCohorts)  ;   DRONES
 [
  set heading 270
  repeat ceiling( number / STEPWIDTHdrones)
  [
    fd 1
    set pcolor color
  ]
  set heading 180
  setxy -5 (- age)
 ]

 ask DroneCohorts
 [
  set heading 270 repeat ceiling( number_healthy / STEPWIDTHdrones)
  [
    fd 1
    set pcolor color
    ]
  repeat ceiling( number_infectedAsPupa / STEPWIDTHdrones)
  [
    fd 1
    set pcolor (color - 2)
  ]
  set heading 180
  setxy -5 (- age - DRONE_EMERGING_AGE - 1)
 ]
end

;
**********************************************************************************
**********************************************************************************
**********



;
================================================================================
================================================================================
=========
; ===============  IBM FORAGING SUBMODEL ========================  IBM FORAGING
SUBMODEL ==================================  IBM FORAGING SUBMODEL
========================
;
================================================================================
================================================================================
=========
```

```
;
*********************************************************************************
*********************************************************************************
**********

;
*********************************************************************************
*********************************************************************************
************

to Start_IBM_ForagingProc
 ; controls the number of foraging trips per day, calls ForagingRoundProc

 let continueForaging true
  ; foraging is continued until it is stopped
 let meanTripDuration 0
 let summedTripDuration 0
 let HANGING_AROUND SEARCH_LENGTH_M / FLIGHT_VELOCITY
  ; [s] duration of a foraging round if all foragers are resting
  ; (= time for unsuccessful search flight)
 let ageLaziness 100
 ; [d] min. age to allow foragers being lazy
 ForagersDevelopmentProc
  ; called before creation of new foragers to avoid ageing by 2d at creation

 NewForagersProc
 ask foragerSquadrons
 [ ; Laziness: lazy bees won't forage and can't be recruited on that day.
  ; applies only to older bees and if the honey store is not too small
  if age >= ageLaziness and
    random-float 1 < ProbLazinessWinterbees and ; ProbLazinessWinterbees: default: 0!
    random-float 1 < (HoneyEnergyStore / DecentHoneyEnergyStore)
   [
    set activity "lazy"
   ]
 ]

 set ForagingSpontaneousProb Foraging_ProbabilityREP
  ; the probability for a resting forager to start spontaneously foraging in a single foraging
  ; round today is calculated in "to-report Foraging_ProbabilityREP "

 set ForagingRounds 0
  ; counter of the foraging rounds
 ask foragerSquadrons
 [
  set activityList [ ]
  ; activityList records all activities of a forager during the day
 ]

 ; always "season" as SEASON_START = 1 & SEASON_STOP = 365
 if ( Day >= SEASON_START )
```

```
    and ( Day <= SEASON_STOP )
     ; foraging takes only place during season and while honey store not
     ; (almost) full (0.95: to avoid foraging, when honey cannot be stored)..
    and
     ( HoneyEnergyStore < 0.95 * MAX_HONEY_ENERGY_STORE
       or PollenStore_g < IdealPollenStore_g )
     ;  ..or when pollen is needed
    and DailyForagingPeriod > 0

[
  while [ continueForaging = true ]
    ; .. and only for a certain time (=DailyForagingPeriod), which is checked
    ; via "continueForaging"
  [
    ask foragerSquadrons
    [
      set activityList lput ForagingRounds activityList
      ; the ForagingRounds is added to a foragers activityList
    ]
    ForagingRoundProc
      ; call ForagingRoundProc, which calls all procedures involved in foraging

    set ForagingRounds ForagingRounds + 1
      ; # foraging rounds is increased

    ifelse ColonyTripForagersSum > 0
      [ set meanTripDuration ColonyTripDurationSum / ColonyTripForagersSum ]
        ; IF > 0 (i.e. if at least 1 foraging trip has taken place):  calculate the average time
        ; a forager needed for its trip in this round
      [ set meanTripDuration HANGING_AROUND ]
        ; ELSE: if no one goes foraging: foraging round lasts "HANGING_AROUND" seconds

    set summedTripDuration ( summedTripDuration + meanTripDuration )
      ; mean trip durations are summed up

    ; if the duration of all foraging rounds summed up is larger than DailyForagingPeriod
    ; then foraging ends for today
    if summedTripDuration >= DailyForagingPeriod
    [
      set continueForaging false
    ]  ; until the total time >= DailyForagingPeriod

    if ((Details = true) and (continueForaging = true))
    [
      if WriteFile = true [ WriteToFileProc ]
    ]
      ; if Details & WriteFile true: results are recorded in Output file after each foraging round (trip)
  ]
]

ForagersLifespanProc
```

```
   ; mortality of foragers due to max. lifespan, max. km or in-hive mortality risk

  ask foragerSquadrons
  [
    set activity "resting"
    set activityList lput "End" activityList
  ]     ; after foraging is completed for today, all foragers do rest
end;

;
*********************************************************************************
*********************************************************************************
************

;  ************** PARAMETERIZATION FLOWER PATCH
************************************************** PARAMETERIZATION FLOWER
PATCH ****************************************

;
*********************************************************************************
*********************************************************************************
************

to CreateFlowerPatchesProc
  ; creates 2  flower patches ("red" & "green"),

  set N_FLOWERPATCHES  2 ; 2
  if readInfile = true
  [
    set bugAlarm true
    show "BugAlarm in CreateFlowerPatchesProc! Check read-in!"
  ]
  create-flowerPatches N_FLOWERPATCHES
  [
    set patchType "GreenField"
    set distanceToColony DISTANCE_G ;1500  ; [m]
    set xcorMap distanceToColony
    set size_sqm 100000
    set quantityMyl  QUANTITY_G_l * 1000 * 1000; [microlitres]
    set amountPollen_g POLLEN_G_kg * 1000 ;10000 ; 10kg = 10000g
      ; total amount of pollen available at this patch

    if SeasonalFoodFlow = true
    [
      set quantityMyl FlowerPatchesMaxFoodAvailableTodayREP who "Nectar"
      set amountPollen_g FlowerPatchesMaxFoodAvailableTodayREP who "Pollen"
    ]

    set nectarConcFlowerPatch CONC_G
      ;  mean nectar concentration returned to colony ca. 1.4 (assessed from Seeley (1986), Fig 2)
```

```
   set detectionProbability DETECT_PROB_G
   set shape "fadedFlower"
   set color green
   set size 4
   ifelse distanceToColony <= 5500
     [ setxy (15.1 + (distanceToColony / 250) )  3 ]  ; IF (distance)
     [ setxy 39.5 3 ]                                 ; ELSE (distance)
  ] ; create-flowerPatches N_FLOWERPATCHES

  ask flowerPatch 0
  [
   set patchType "RedField"
   set distanceToColony DISTANCE_R  ; [m]   ; RED PATCH
   set xcorMap -1 * distanceToColony
   set quantityMyl QUANTITY_R_l * 1000 * 1000 ; [microlitres]
   set amountPollen_g POLLEN_R_kg * 1000 ; [g]

   if SeasonalFoodFlow = true
   [
     set quantityMyl FlowerPatchesMaxFoodAvailableTodayREP who "Nectar"
     set amountPollen_g FlowerPatchesMaxFoodAvailableTodayREP who "Pollen"
   ]

   set nectarConcFlowerPatch CONC_R
   set detectionProbability DETECT_PROB_R
   set color red

   ifelse distanceToColony <= 5500
    [ setxy (14.9 - (distanceToColony / 250) )  3 ]
    [ setxy -7.5 3 ]
  ]

  FlowerPatchesUpdateProc
end;

;
*********************************************************************************
*********************************************************************************
***********

;  ************** PARAMETERIZATION FLOWER PATCHES FROM FILES
*********************************************************************************
*********************************

;
*********************************************************************************
*********************************************************************************
************

to Create_Read-in_FlowerPatchesProc
  ; copy of CreateFlowerPatchesProc but data are read from input file
```

```
; calculates derived values (e.g. EEF, flight costs etc)

let counter 0
set TodaysAllPatchesList []
  ; shorter list, contains data of all patches, but only for today

set TodaysSinglePatchList []
  ; short list, contains data of a single patch for today

set counter Day
  ; counter: to chose only the values for today from the complete
  ; (all days, all patches) list

repeat N_FLOWERPATCHES
[
  ; todays data for ALL N_FLOWERPATCHES flower patches are saved in a
  ; new, shorter list (= todaysAllPatchesList)

  set TodaysSinglePatchList (item counter AllDaysAllPatchesList)
    ; this new, shorter list (= todaysAllPatchesList) is comprised of very
    ; short lists (=todaysSinglePatchList) that contain only the data of the
    ; current patch and only for today

  set todaysAllPatchesList fput TodaysSinglePatchList todaysAllPatchesList
    ; fput: faster as lput! (Netlogo 4) however: list is in reversed order!

  set counter counter + 365
  create-flowerPatches 1
  [
    set oldPatchID item 2 TodaysSinglePatchList
      ; refers to patch number of crop maps from a landscape module,
      ; an optional external tool to read in and analyse maps of food patches

    set patchType item 3 TodaysSinglePatchList ; e.g. Oilseed rape
    set distanceToColony item 4 TodaysSinglePatchList  ; [m]
    set xcorMap item 5 TodaysSinglePatchList ; x coordinate
    set ycorMap item 6 TodaysSinglePatchList ; y coordinate
    set size_sqm item 7 TodaysSinglePatchList  ; patch area [m^2]
    set amountPollen_g item 8 TodaysSinglePatchList ; [g]
    set nectarConcFlowerPatch item 9 todaysSinglePatchList  ; [mol/l]
    set quantityMyl  (item 10 TodaysSinglePatchList) * 1000 * 1000 ; [microlitres]

    let calcDetectProb item 11 TodaysSinglePatchList
    ; calculated in "2_BEEHAVE_FoodFlow"-Tool on basis of distance
    ; (if this input file is created by "BEEHAVE_FoodFlow")

    let modelledDetectProb item 12 TodaysSinglePatchList
      ; modelleded in "3_BEEHAVE_LANDSCAPE" with individual scouts
      ; exploring a 2-dim landscape

    ifelse ModelledInsteadCalcDetectProb = true
```

```
      [ set detectionProbability modelledDetectProb ]
      [ set detectionProbability calcDetectProb ]

    set shape "flower"
    set size 1 + (sqrt size_sqm) /  1000
    setxy (distanceToColony / 300) 3
   ]
 ]  ; END of "repeat N_FLOWERPATCHES"
 FlowerPatchesUpdateProc
 set TodaysAllPatchesList reverse TodaysAllPatchesList
  ; to correct the reversed order, caused by the fput command

end;

;
**********************************************************************************
**********************************************************************************
**********

to FlowerPatchesUpdateProc
 let energyFactor_onFlower 0.2 ; (0.2)
  ;  reflects reduced energy consumption while bee is sitting on the flower
  ; to collect nectar or pollen;
  ; Kacelnik et al 1986 (BES:19): 0.3 (rough estimation, based on Nunez 1982)

 ;                                                HANDLING TIME:
 ask flowerPatches
 [
  if ReadInfile = false
  [
   ifelse ConstantHandlingTime = true
    [
     set handlingTimeNectar TIME_NECTAR_GATHERING  ; IF: handling time constant
     set handlingTimePollen TIME_POLLEN_GATHERING
    ]
    [
     if quantityMyl > 0
     [
      set handlingTimeNectar
       TIME_NECTAR_GATHERING *
       ((FlowerPatchesMaxFoodAvailableTodayREP who "Nectar") / quantityMyl)
     ] ; ELSE: handling time dependent on proportion of nectar or pollen left

     if amountPollen_g > 0
     [
      set handlingTimePollen TIME_POLLEN_GATHERING
      * ((FlowerPatchesMaxFoodAvailableTodayREP who "Pollen") / amountPollen_g)
     ]
    ]
  ] ;    if ReadInfile = false
```

```
if ReadInfile = true
[
 set TodaysSinglePatchList item who TodaysAllPatchesList
 ifelse ConstantHandlingTime = true
  [ ; IF CONSTANT handling time:
    set handlingTimeNectar item 13 TodaysSinglePatchList
     ;  item 13: handling time nectar
    set handlingTimePollen item 14 TodaysSinglePatchList
  ] ;   item 14: handling time pollen
  [
    ; ELSE: if handling time is NOT constant:
    if quantityMyl > 0   ; nectar handling time
    [
      set handlingTimeNectar (
        item 13 TodaysSinglePatchList) *
        ((item 10 TodaysSinglePatchList) * 1000 * 1000) / quantityMyl
    ] ; item 13: NectarGathering_s, item 10: quantityNectar_l

    if amountPollen_g > 0   ; pollen handling time
    [
      set handlingTimePollen
      (item 14 TodaysSinglePatchList)
      * ((item 8 TodaysSinglePatchList) / amountPollen_g)
    ]  ; item 14: PollenGathering_s; item 8: quantityPollen_g
  ]
] ; if ReadInfile = true

 ;                                                FLIGHT COSTS & EEF:
set flightCostsNectar
 ( 2 * distanceToColony * FLIGHTCOSTS_PER_m)
  + ( FLIGHTCOSTS_PER_m * handlingTimeNectar
  * FLIGHT_VELOCITY * energyFactor_onFlower ) ; [kJ] = [m*kJ/m + kJ/m * s * m/s]

set flightCostsPollen
 ( 2 * distanceToColony * FLIGHTCOSTS_PER_m)
  + ( FLIGHTCOSTS_PER_m * handlingTimePollen
  * FLIGHT_VELOCITY * energyFactor_onFlower )

set EEF ((nectarConcFlowerPatch * CROPVOLUME
      * ENERGY_SUCROSE) - flightCostsNectar) / flightCostsNectar
 ;  Energetic Efficiency of the flowerPatch

    ;                                                TRIP DURATION:
set tripDuration 2 * distanceToColony * (1 / FLIGHT_VELOCITY )
 + handlingTimeNectar
 ; duration of nectar foraging trip depends on speed, 2*distance + time to
 ; collect nectar from the flowers

set tripDurationPollen 2 * distanceToColony
 * (1 / FLIGHT_VELOCITY ) + handlingTimePollen
 ; duration of pollen foraging  trip depends on speed, 2*distance + time to
```

```
    ; collect pollen from the flowers
        ;                                                MORTALITY:
   set mortalityRisk 1 - ((1 - MORTALITY_FOR_PER_SEC) ^ tripDuration)   ; nectar foragers
   set mortalityRiskPollen 1 - ((1 - MORTALITY_FOR_PER_SEC) ^ tripDurationPollen) ; pollen foragers
        ;                                                 DANCING:
   set danceCircuits DANCE_SLOPE * EEF + DANCE_INTERCEPT  ; derived from Seeley 1994

   if danceCircuits < 0 [ set danceCircuits 0 ]
   if danceCircuits > MAX_DANCE_CIRCUITS [ set danceCircuits MAX_DANCE_CIRCUITS ]
    ; MAX_DANCE_CIRCUITS: ca. 100 (Seeley, Towne 1992)

   if SimpleDancing = true
   [
    ifelse EEF > 20
      [ set danceCircuits 40 ] ; IF
      [ set danceCircuits 0 ]  ; ELSE
   ]

   if AlwaysDance = true [ set danceCircuits 40 ]
    ; in this case, foragers always dance for their patch,
    ; irrespective of its quality

   set danceFollowersNectar danceCircuits * 0.05
    ; Seeley, Reich, Tautz (2005): "0.05 recruits per waggle run (see Fig. 3)"
  ] ; ask flowerPatches
end

;
*******************************************************************************
*******************************************************************************
************

to ForagingRoundProc
  ;  CALLED BY Start_IBM_ForagingProc calls Procedures involved in each foraging trip
  ; and does foraging related plots

  set ColonyTripDurationSum 0
  set ColonyTripForagersSum 0
   ; used to calculated duration of this foraging round
  set DecentHoneyEnergyStore (TotalIHbees + TotalForagers ) * 1.5 * ENERGY_HONEY_per_g
   ; DecentHoneyEnergyStore reflects the amount of energy a colony should store
   ; to survive the winter, based on the assumption that a bee consumes ca. 1.5g honey during winter
  if DecentHoneyEnergyStore = 0
  [
    set DecentHoneyEnergyStore 1.5 * ENERGY_HONEY_per_g
  ] ; to avoid division by 0

  ; Proportion of pollen foragers:
  set ProbPollenCollection (1 - PollenStore_g / IdealPollenStore_g)
    * MAX_PROPORTION_POLLEN_FORAGERS
    ; (Pollen foragers: ~ 0-90% of all foragers: Lindauer 1952)
```

```
if HoneyEnergyStore / DecentHoneyEnergyStore < 0.5
[
  set ProbPollenCollection ProbPollenCollection
  * (HoneyEnergyStore / DecentHoneyEnergyStore)
]

FlowerPatchesUpdateProc
Foraging_start-stopProc  ; some foragers might spontaneously start foraging
Foraging_searchingProc    ; unexperienced foragers search new flower patch
Foraging_collectNectarPollenProc  ; succesful scouts and experienced Foragers gather nectar
Foraging_flightCosts_flightTimeProc  ; energy costs for flights and trip duration
Foraging_mortalityProc  ; foragers might die on their way back to the colony
Foraging_dancingProc    ; successful foragers might dance..
Foraging_unloadingProc  ; ..and unload their crop & increase colony's honey store

let foragersAlive SQUADRON_SIZE * count foragerSquadrons

let currentNectarForagers
   SQUADRON_SIZE * count foragerSquadrons with
     [activity = "expForaging" and pollenForager = false]

let currentPollenForagers
   SQUADRON_SIZE * count foragerSquadrons with
      [activity = "expForaging" and pollenForager = true]

let currentResters
   SQUADRON_SIZE * count foragerSquadrons with
      [activity = "resting"]

let currentScouts
   SQUADRON_SIZE * count foragerSquadrons with
      [activity = "searching"]

let currentRecruits
   SQUADRON_SIZE * count foragerSquadrons
      with [activity = "recForaging"]

let currentLazy
   SQUADRON_SIZE * count foragerSquadrons
      with [activity = "lazy"]

if sqrt ((foragersAlive - currentNectarForagers ; to avoid bugAlarm caused by numeric inaccuracy!
  - currentPollenForagers - currentResters
  - currentScouts - currentRecruits - currentLazy) ^ 2) > 0.0000000001
[
  set BugAlarm true show "BUG ALARM in ForagingRoundProc: wrong number of forager activities!"
]

if ShowAllPlots = true
[
```

```
let i 1
while [ i <= N_GENERIC_PLOTS ]
[
 let plotname (word "Generic plot " i)
  ; e.g. "Generic plot 1"

 set-current-plot plotname
 if (i = 1 and GenericPlot1 = "active foragers today [%]")
 or (i = 2 and GenericPlot2 = "active foragers today [%]")
 or (i = 3 and GenericPlot3 = "active foragers today [%]")
 or (i = 4 and GenericPlot4 = "active foragers today [%]")
 or (i = 5 and GenericPlot5 = "active foragers today [%]")
 or (i = 6 and GenericPlot6 = "active foragers today [%]")
 or (i = 7 and GenericPlot7 = "active foragers today [%]")
 or (i = 8 and GenericPlot8 = "active foragers today [%]")
 [
  create-temporary-plot-pen "active%"
  set-plot-y-range  0 110
  set-plot-pen-mode 0 ; 0: lines
  ifelse foragersAlive > 0
   [
    plot (100 * SQUADRON_SIZE ; % active foragers of all foragers CURRENTLY alive
          * (count foragersquadrons
               with [ activity != "resting" and activity != "lazy"] )) / foragersAlive
   ]   ; i.e. with activities = "searching", "recForaging" or "expForaging"
   [
    plot 0
   ]
  create-temporary-plot-pen "deaths%"
  set-plot-pen-color red
  plot 100 * DeathsForagingToday ; cumulative deaths as % of todays' INITIAL foraging force
       / (foragersAlive + DeathsForagingToday)
 ]

 if (i = 1 and GenericPlot1 = "foragers today [%]")
 or (i = 2 and GenericPlot2 = "foragers today [%]")
 or (i = 3 and GenericPlot3 = "foragers today [%]")
 or (i = 4 and GenericPlot4 = "foragers today [%]")
 or (i = 5 and GenericPlot5 = "foragers today [%]")
 or (i = 6 and GenericPlot6 = "foragers today [%]")
 or (i = 7 and GenericPlot7 = "foragers today [%]")
 or (i = 8 and GenericPlot8 = "foragers today [%]")
[
 create-temporary-plot-pen "nectar"
 set-plot-pen-color yellow
 set-plot-pen-mode 0 ; 0: lines
 set-plot-y-range 0 100
 ifelse foragersAlive > 0
  [   plotxy ForagingRounds (100 * currentNectarForagers) / foragersAlive
    create-temporary-plot-pen "pollen"
     set-plot-pen-color orange
```

```
        plotxy ForagingRounds (100 * currentPollenForagers) / foragersAlive
      create-temporary-plot-pen "scouts"
        set-plot-pen-color green
        plotxy ForagingRounds (100 * currentScouts) / foragersAlive
      create-temporary-plot-pen "resters"
        set-plot-pen-color brown
        plotxy ForagingRounds (100 * currentResters) / foragersAlive
      create-temporary-plot-pen "lazy"
        plotxy ForagingRounds (100 * currentLazy) / foragersAlive
      create-temporary-plot-pen "recruits"
        set-plot-pen-color blue
        plotxy ForagingRounds (100 * currentRecruits) / foragersAlive
    ]
    [
      plotxy ForagingRounds 0
      create-temporary-plot-pen "pollen"
        set-plot-pen-color orange
        plotxy ForagingRounds 0
      create-temporary-plot-pen "scouts"
        set-plot-pen-color green
        plotxy ForagingRounds 0
      create-temporary-plot-pen "resters"
        set-plot-pen-color brown
        plotxy ForagingRounds 0
      create-temporary-plot-pen "lazy"
        plotxy ForagingRounds 0
      create-temporary-plot-pen "recruits"
        set-plot-pen-color blue
        plotxy ForagingRounds 0
    ]
  ]  ; END:  if plotChoice = "foragers today [%]"

  set i i + 1
 ]

 ] ; if ShowAllPlots = true
end

;
*******************************************************************************
*******************************************************************************
************

to ForagersDevelopmentProc
 ; foragers age by 1 day, forager turtles move forward
 ask foragerSquadrons
 [
  set age age + 1
  fd 1.8   ; movement on GUI
 ]
end
```

```
;
**********************************************************************************
**********************************************************************************
************

to NewForagersProc
 ; creates foragerSquadrons as turtles, based on # in-hive bees developing into foragers

 let foragerSquadronsToBeCreated
   NewForagerSquadronsHealthy
   + NewForagerSquadronsInfectedAsPupae
   + NewForagerSquadronsInfectedAsAdults
 let newCreatedBees 0

 create-foragerSquadrons foragerSquadronsToBeCreated
 [
  set newCreatedBees newCreatedBees + 1
  ifelse ticks = 1
   [
    set age 100 + random 60 ; age of initial foragers: 100d + 0..59d
    setxy 40 9
    set color grey
    set size 2
    set heading 90
    set shape "bee_mb_1"
    set mileometer random (MAX_TOTAL_KM / 4)
   ]  ; IF 1st time step: (=initial bees): travelled distace equally distributed,
     ; (winterbees have done only little foraging in autumn!)

   [ ; ELSE: all other foragers
    set age Aff
    setxy (-20 + age) 9
    set color grey
    set size 2
    set heading 90
    set shape "bee_mb_1"
   ]

  set activity "resting"
  set activityList [ ]
  set cropEnergyLoad 0 ; [kJ] no nectar in the crop yet
  set collectedPollen 0 ; [g] no pollen pellets
  set knownNectarPatch -1  ; -1 = no nectar Flower patch known
  set knownPollenPatch -1 ; -1 = no pollen Flower patch known
  set pollenForager false
   ; foragers are nectar foragers except they decide to collect pollen

  ; creation of HEALTHY and INFECTED foragers:
  set infectionState "healthy"
   ; possible infection states are: "healthy" "infectedAsPupa" "infectedAsAdult"
```

```
    if newCreatedBees > NewForagerSquadronsHealthy
    [
      set infectionState "infectedAsPupa"
      set ycor ycor - 3
    ]  ; foragers infected as pupa are created

    if newCreatedBees > (NewForagerSquadronsHealthy + NewForagerSquadronsInfectedAsPupae)
    [
      set infectionState "infectedAsAdult"
      set ycor ycor + 1.5
    ] ; foragers infected as adults are created
  ] ; create-foragerSquadrons

  ; the toal number of ever produced foragers is recorded and can be used as output:
  set SummedForagerSquadronsOverTime
     SummedForagerSquadronsOverTime
     + NewForagerSquadronsHealthy
     + NewForagerSquadronsInfectedAsPupae
     + NewForagerSquadronsInfectedAsAdults

  ; no more new foragers have to be created in this time step:
  set NewForagerSquadronsHealthy 0
  set NewForagerSquadronsInfectedAsPupae 0
  set NewForagerSquadronsInfectedAsAdults 0
end;

;
*******************************************************************************
*******************************************************************************
************

to-report Foraging_PeriodREP
  let foragingPeriod_s -1
  let foragingHoursList [ ]
  ; "foragingPeriod" = HOURS SUNSHINE ON DAYS WITH Tmax > 15degC

  ; 2000: from weather data Berlin, Germany (DWD), (1.1.-31.12.2000);
  let foragingHoursListBerlin2000
    [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7.2
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.5 0 0 0 0
      0 0 0 10.7 0 0 0 0 0 0 0 0 0 0 0 7 0 7.9 6.8 4.7 10.8 11.2 11.8
      11.2 9.9 0 10.7 10.4 4.2 10.6    8.7 5.7 13.3 13.2 12 14 14.1 13.9
      13.1 10.7 7.1 13.7 14.6 15 15.1 15 13.5 10.3 2.6 5.9 0 6 0 8.4 2.4
      0.7 12.1 5.8 6.8 8.7 6 10 8.7 14.2 12.3 7.4 3.4 0.2 7.2 13.2 15.8
      13.9 9.5 11 15.3 4.1 2.1 6 12.7 10.4 15.4 15.1 11.4 8.5 8 1.5 1.5
      2.4 2.6 1.1 0.1 0 9.5 4.5 2.4 3.9 1.3 2.2 8.3 1.1 3.4 2.8 5.1 0.2
      6.4 0.5 3.4 5.2 5.4 0.1 0 1.5 0 0.5 7.9 9.8 4.4 1.6 3.8 2.1 0.6 1
      1.5 10.7 3.8 8.3 7.1 9.3 12.7 6.9 3.6 10.3 3.3 0.2 5.7 11.7 13.4
      7.8 5.2 9.5 5 4.2 5.4 2 7.3 8.5 9 4.7 13.1 10.5 0 7.5 8.6 4.3 8
```

2.5 0 2.2 1.2 8.1 2.8 0 0.4 5.1 1.2 6.2 2.1 0.1 5.1 0.3 0 11.7 0
0 10.4 6.5 11.1 11.3 8.5 1.2 8.8 5.6 10.6 10.3 8.1 3.7 9.4 2.2 0.2
0 0 0 0 0 2.2 2.9 2.7 6.9 0 6 3.3 0 0 0 7.4 9.1 8.9 1.7 0 0 0 0 4.1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

; 2001:
let foragingHoursListBerlin2001
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.3 10.3 6.2 5.5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 8.1 3.9 6.6 0 3 10.9 13
13.2 13.6 4.9 0 0 0 9 14.2 14.2 14.7 13.7 12.2 12.6 2.1 8.3 2.9 5.3
10.1 13.1 8.3 7.5 15.3 15.1 14.9 11.6 6.5 0 6.2 3.5 1 2 0 0 0.7 1.2
3.1 3.1 1.4 8.9 0 6.9 0 11.3 4.6 6.8 4 8.5 3.2 5.7 14.3 3.3 3.3 2.5
6 13.6 13.3 14.3 1.7 10.6 12.8 5.6 0.9 12.6 12.4 11.2 13.1 6.6 0.4
0 5.5 5.4 11.1 6.5 2.5 3 0 0.6 8.5 11.9 11.2 5.9 11.1 7.9 11 10.4
10.9 14.9 14.5 6.3 12.2 2.7 5.8 12.6 3.9 2.8 5.2 6.5 5.3 5.9 8.5 7.3
7.4 1.1 0 5.6 13.3 12.8 6.2 0 2.9 6.6 0 9.3 11.8 8.3 10.3 11 3.8 4
4.3 10.9 2.9 3.9 2.5 0.3 1.2 8.1 2.9 1.6 6.2 0 0.2 0 2.1 0.2 1.5 4.2
3.8 3.5 0 9.9 0.5 2.6 1.1 9 0 0 0 0 0.8 4.3 0 0 0 2.2 4.5 3.8 9.5 1.1
7.9 3.9 7.6 0 7.7 7.5 6.3 1.2 5.5 0 0 1.9 6.9 0 0 0 0 0 5.7 0 0 0 3.1
2.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

; 2002:
let foragingHoursListBerlin2002
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 5.9 8.3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9.7
0 0 0 0 0 0 7.2 0 0 0 0 0 0 0 0 0 0 11.2 9.1 2.8 11.2 11 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 5 0 6.7 13.3 3.3 0 0.2 3.2 0 0 0 0 2.7 1 5.8 0 0 4.5 0 8.1
12.7 11.7 5.2 5.6 7.9 6.7 4.3 10.4 13.7 14.7 0 8.6 10.9 12.9 7.7 2.4 1.4 0
6.1 0 6.7 11.3 6.1 10.3 13.3 10.4 8.9 7.7 3.9 0 0 0.4 1.7 4.6 1.3 0.2 3
4.8 6.2 11.1 14.4 6.4 6 4.3 9.9 6.3 9 10.3 10.1 7.4 8.3 5 1.4 0 2 1.9 0.3
12.2 5.7 4.5 12.9 14.5 11.5 8.2 6.9 7.8 0 1.4 6.4 0.9 0.6 0 2.9 11.7 0.9
1.6 2 2.9 0.4 8.6 14.3 11.3 11.5 7.1 7.6 0.7 13.4 8.8 0.1 7.5 4.3 2.9 3.7
4.7 9.1 0 0 1.2 10.4 6.1 6.3 12.2 12.3 12.9 11.8 9.2 10.7 9 9.3 10.6 10.8
10.5 8.5 8.6 6.7 7.8 11.8 10.4 10.6 6.7 10.6 4.8 10.4 10.9 9 7.2 12.1 10.2
3.7 8.8 1.5 1.9 3.3 4.3 0.3 2.6 0 0 0 9.4 0 0 0 0.7 6.6 9.3 8.9 6.2 4.3 0
0 0 0 0 0 0 0 0 0 1.2 0 0 0 0 0 0 0.9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

; 2003:
let foragingHoursListBerlin2003
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 11.5 7.9 0 1.8 0 9.6 8.1 0 9.7 0 0 0 0 0 0 0 0 0 0 11.7
12.4 12.4 12.5 12.7 0 0 11.8 11.7 12.8 12.4 8.6 8.6 0.1 7.3 4.7 2.9 4.5 7.3
9.5 3.1 13.5 12.4 7.7 9.4 11.6 0.5 4.9 10.6 4.1 3.1 4.6 0 5.3 6.7 7.3 1.7
5.5 5.9 8.1 1.1 13.1 14.3 5.6 10.3 9.9 15.4 15.4 7.8 14.3 14.4 12.5 13.6 10.9

11.4 13.6 13.2 11.2 13.6 9.3 12.4 12.5 8.8 8.9 10.3 13.3 3.6 1.8 5.3 2.8 10.8
5.7 10.9 2.7 3.8 13.9 15.2 5.2 11.6 2.3 6.1 8.1 1.3 0.4 0.1 3.6 4.5 3.1 6.2
13.4 4.2 6.4 15.7 13.3 13.2 4 6.5 13.4 13.3 8.5 12.6 8.9 6.6 4.2 2.2 7.6 5
7.5 12.6 4.6 10.4 5 8.1 12.8 12.8 12.1 13.9 13.8 13.9 14.2 14.4 10.5 13 4.6
9.9 9.4 13.3 6 3.6 10.1 9.3 9.4 4.3 6.8 11.9 7.2 2.6 2.7 2.3 4.6 7.8 3.8 10.8
2.7 0.8 11.7 11.2 5.7 9.7 2 3.5 0 1.3 3.5 6.1 10.8 8.2 6.9 10.7 11.4 11.3 11.6
11.1 2.8 9.6 11.4 11.3 3.1 6.5 2.4 0 9.6 1.7 2.4 3.1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

; 2004:
let foragingHoursListBerlin2004
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 9.9 7 1.3 0 0 0 0 0 0 0 0 0 0 12 10.6 0 6.5 2.5 0 0 0 0 0 0 0 0 13.2 13.1
12.5 11 8.4 0.5 4.5 3.2 10.4 0 0 10.2 0 2 13.9 11.6 12.5 7.2 0 3.9 5.2 8.2 3.9
2.3 0 0 4.4 4.7 0 0 6.6 0 9.6 0.4 8 8.4 8.1 0 0 0 0 1.6 0 0 11.6 15.2 14.8 8.7
7.9 0 12.8 4.2 1.1 12.1 8.2 9.4 2.9 4.6 4 9.1 6.2 6.6 5.5 9.6 1 2.6 4.9 11.7
11.6 7.7 4.9 5.2 5.4 6.3 0.2 8.6 8.1 4.5 5.8 9.3 7 7.5 6 11.4 13.7 4 3.6 3.9 9.6
1 0.8 4.2 2.5 1.1 7.5 10.4 7 9.6 5 3.3 10.3 6.5 6.4 4.1 6.7 11.2 14.8 14.4 11.5
9.7 8.3 8.5 12.2 11.9 13.9 12.4 12.6 12.9 13.7 7.3 11.5 4.9 5.2 12 7.5 5.1 6.3
6.2 4.2 5.8 10.1 7.1 2.7 2.9 3 1.9 2.1 3.2 0.7 3.8 6.7 12.2 12.4 12.4 12.8 12.4
10.7 11.6 12.6 12.5 4.5 5.1 4 5.2 7.8 8.1 11.6 11.7 4.7 2.4 1.5 3.2 0 0 3.9 0
0.2 0.8 1 3.5 0.8 3.7 8.7 5.3 9.5 1.9 8.1 0 0 0 0 0 0 0 0 0 0 0 6.5 9.3 1.3
5.4 3.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

; 2005:
let foragingHoursListBerlin2005
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.8 0.4
0 0 0 0 0 4.7 7 0 0 0 0 0 0 12.3 10.4 11.7 0 9.4 0 0 0 0 0 6.3 2.1 6.7 7 10.2
10 11.4 10.3 0 0 0 0 0 4.4 11.7 0 8.4 8.5 6.6 11.7 9.7 5 2.5 7.1 2.3 0 0 0 0 0
0 11.8 1.6 0 8.4 0 0 12.7 11 5.7 4.7 0.4 5.4 9.6 12.7 13.9 15 14.2 0 4.3 0 2.8
7.9 6.7 2.5 0 0 9.5 6.6 1.2 0 0 11.7 10.2 7.9 11.5 0.4 14.1 11.1 16 11.9 7.2 15.7
9.8 8.7 14.8 15.7 15 13.8 10.9 0.1 3.2 9.2 12 0 1.1 2.1 0.1 3 14.3 14.8 14.9 13.7
12 11 9.1 7.3 6.4 4.7 4.3 0 3 0.2 4.6 4 2.1 6.8 7.9 6.8 6.9 9.4 8.5 10.1 0 6.4
5.6 3.9 5.1 11.1 0.5 0 1.3 8.4 0.6 1.2 4 10.9 6.6 13.7 12.4 8.4 11.5 11.1 0 6.5
0.2 5.6 11.3 10 12.8 12 12.8 12.3 8.4 0.9 12.4 12.4 12.5 11.9 11.7 11.7 7.4 0 0.2
6.6 6.9 7 0 8.1 11.7 6.8 5 0.7 11.3 11.2 10.3 10.5 3.6 7.4 0.8 0 3.4 1.7 0 0 5.4
9.5 10 9.4 8.9 9.2 7.5 9.8 9.7 9.2 9.6 8.6 0 0 0 0 5.7 0 0.2 2.2 0 0 3.3 7.7 8.9
8.6 8.2 0 0 0 2.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

; 2006:
let foragingHoursListBerlin2006
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1.8 1.5 5 0 0 0 3.8 8.5 0 0 0 0 0 8.2 0 0 0 0 0 0 0 0.1 0 0 6.5 9.6 4.3
0 3.7 0 13.1 4.7 0 0 0 0 10.5 5.6 13.4 12.5 11.9 11 12.2 10.6 14.2 14.7 14.1 12.6
6.8 4.6 10.5 8.6 1.4 0.3 3.5 6.1 1.5 7.7 5.8 9.9 0 0 1.6 6.6 0 0 2.4 0 11.5 4.4 0

```
0 4.8 9 11.5 11.5 15.6 15.8 15.8 15.7 15.2 7.5 5.6 1.2 9.1 9.8 9 7.7 6.4 9.8 12.4
13 9.7 12.3 10.4 10.2 0.7 14.2 15.8 16 16 15.7 12.9 10.6 2.5 12.3 11.7 10.8 13.3
8.5 10.3 11.5 13.4 15.7 15.7 15.5 13.9 12 14.1 6 14 12.9 14.8 13.6 5 5 12.9 6 9.3
8.5 6.4 3.5 0.6 0.8 9.3 4.6 5.3 2 3.9 8.4 0 9.8 2.2 6.9 8.2 3.7 11.2 7.7 4.9 7 0.9
9.6 3.5 2.3 4.2 6.7 1.2 0.2 4.2 0.2 7.7 0 5.1 9.1 3.7 8.5 6.4 5.3 11.9 12.4 11.5
12.1 12 11.4 6.4 4.7 9.2 1 8.9 11.3 11.5 11.4 11.3 11.1 9.5 0.1 3 10.2 7.8 3.9 1.3
0.4 0.2 2.9 0.9 1.4 4.2 9.8 9.1 6.3 8.2 0 0 0 6.7 9.9 7.9 4.8 0 6 5.3 3.2 2.7 4.4
6.3 7.1 0 1.1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.7 7.7 1.3 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]

; ROTHAMSTED WEATHER DATA 2009:
;TH: 15C:
let foragingHoursListRothamsted2009 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 10.4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7.8 0 0 8.9 0 5.4 0 0 0 0 0 0 4.1 6 5.9
0 0 0 0 10.1 12.3 11 9.3 10.5 0 11.5 0 0 11.2 4.5 8 10.3 0 0 5.2 7.5 3.2 0 9.4 10.3 0 11.6
0 0.7 0 0 0 6.9 5.4 8.2 8.7 8.4 12.5 15 7.5 7.5 0.7 6.7 13 15 14.2 14.3 14.9 3.4 11.7 0 0
4.3 2.5 0 0.9 6.5 11.8 5.4 13 5.4 9.4 4.7 6 9.7 2.7 9 5 10.6 13.9 8 2.7 4.7 4.3 10.8 11.7
12.7 12.3 6.2 11.8 9 6.8 4.7 3.7 5.2 9.7 2.2 7.4 7.4 8.7 6.1 3.6 1.9 5.3 3.8 7.8 0.2 7.1 6.1
6.5 11.4 1.8 5.1 6.8 1.6 8.7 8.6 0.9 8.5 5.4 0 5.9 3.2 2.7 9.5 4.8 2.7 8.5 1.8 6.2 3.2 2.6
10.4 7.5 7.5 12.3 5.4 8.4 8.1 11.4 7.3 5.8 2.3 7.4 7.4 8.7 3.8 5.7 7.3 0.4 5.2 7.5 6.1 4.3 0.5
6.7 5.7 7 4.8 9.8 0.8 3.6 0 4.6 1.6 7.7 3.4 4.4 4.9 3.3 1.8 9.7 9.9 8 9.3 0.9 5.2 0.3 5.6 5.5
0.8 4.9 0.1 0.1 0 0 0 4 3.5 0 0 0 0 0 0 0 0 0 0 0 0 6.2 0.5 4.2 0 1.3 0.6 1.8 0 2.5 0.5 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]

; ROTHAMSTED WEATHER DATA 2010:
; TH: 15C:
let foragingHoursListRothamsted2010 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 5.3 0 0 0 0 0 5.8 0 0 0 0 0 0 0 0 0 0 0 0 0 9.3 0 11.4 9.1 10.6 0 0 0 0 0
0 11.8 11.4 0 0 0 0 13.1 11.2 2.4 4.4 10.4 8.1 1.7 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8.3 0
6.4 7 8.1 5.9 12.5 14.9 15 14.7 9 5 6.2 10.7 0 10.1 1.1 0 12.8 15.4 12.9 8.5 3.7 5.7 3.1 2.8
0.9 5 4.5 5 6.5 9 12.1 13.9 1.5 0 6.9 9.1 14.6 13 10.2 9 8.9 13.7 14 6.2 7.6 7.3 3.8 10.3
10.2 7.2 7.6 1.4 6.5 12.5 10.8 7.3 4.6 0 2.2 4.1 6.8 9.6 6.3 9.3 5.8 10.3 7.6 1.7 7 2.9 0.9
1.2 2 2 4 6 1.3 3.3 7.3 0.8 5.8 4.6 3 5.8 9.3 1.1 9.6 2.9 2.8 1.4 8 7.2 2.1 6.6 4.9 1.1 1.3
6.3 2.9 8 1.6 0 2.9 7.5 4.7 6 10.2 11.3 11.1 8.5 5.6 2.4 4 5 1.6 4.2 1 3 8.6 2.3 0 5 4.6 6.3
7.3 1.1 5.2 7.5 8.7 1.3 0 0 0 0 0.3 0.1 6.6 0 3.1 1.3 0.1 0.7 5.2 6 4.1 0 6.4 8.6 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 4.7 0 0 0 0 0 0 3.3 2.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]

; ROTHAMSTED WEATHER DATA 2011:
; TH: 15degC
let foragingHoursListRothamsted2011 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5.4 1.7 0 0 0 0 0 0 0 0 0 0 0
0 0 3.2 0 4.9 0 0 3.9 0 0 0 0 0 3.1 1.1 10.1 10.5 10.2 3 7.5 6.5 4.6 0.2 4.8 3.3 3.9 6.1 6.2
0 11.5 10.2 12.5 12 11.1 8.5 10.2 0 0.2 5 3.6 6.6 6.7 11.3 8.5 7.8 12.6 12.7 10.1 12.8 4.6
10.9 6.8 5.3 12.9 12.2 13 13.2 13.6 6.6 11.8 3.2 6.8 10.8 11 2.1 8 7.2 8.7 5.1 3.6 2.6 1.3
9.8 8.6 12.3 9.4 4.5 11.9 13 3.8 4.1 2.9 4.2 1.5 10.8 9.7 9.8 13.5 10.7 2.6 0.9 9.1 8.5 4.5
6.6 9.4 2 6.6 11.8 3.6 5.2 1.3 6.7 9.8 7.1 7.1 5.2 7 7.3 6.7 12 8.9 1.6 11.1 8.2 8.5 8.3 4.8
4.6 8.7 6.7 4.4 3.3 5.6 4.2 8.3 1 2.1 8.1 9.5 3.2 3.1 1.1 3.6 1.4 1.3 8 6.6 12.7 9.2 1.7 2.3
```

```
6.9 2.2 11.3 8.7 7.5 6.9 8.7 0.3 3.5 1.8 4.9 7.5 10.1 7.1 2.5 2.8 2 6.4 7.2 3.5 4.1 0.1 9.6
5.4 6.6 8.8 0 4.2 3.2 1.2 5.6 4.4 4.6 0 1 6.2 8.4 5 2 5.8 0 1.2 1.5 1.5 2.3 5.2 6.9 7.5 8.6 7
4.9 5.9 6 6.6 0.2 2.6 5.3 8.4 6.4 6.9 2.7 6.3 9.5 9.7 9.8 9.2 9.7 6.3 4.1 1.3 7 3.3 0.8 2.3 4.7
1.6 2.3 0.1 8.3 9.3 4.5 2.3 8.2 6 0 3.3 9.1 6.4 4.8 2.8 5.8 0 8.3 4.1 0 0 4.1 3.5 0 1.4 0.5 0
0 0 1.1 1.2 0 0.7 5.9 0 0 0 2.2 3 0 0 0 0 0 2 0 2.8 5.6 0 0.3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.4 0 0 0 0 0 ]

 if Weather = "Rothamsted (2009-2011)"
 [
  let inputYear 2011 + round ((ceiling (ticks / 365)) mod 3)
  if inputYear > 2011
  [
    while [inputYear > 2011] [ set inputYear inputYear - 3]
  ] ; after 3 years, 1st dataset is used again etc.

 ;if day = 1 [ type "Rothamsted weather data, year: " print inputYear ]
  if inputYear = 2009 [ set foragingHoursList foragingHoursListRothamsted2009 ]
  if inputYear = 2010 [ set foragingHoursList foragingHoursListRothamsted2010 ]
  if inputYear = 2011 [ set foragingHoursList foragingHoursListRothamsted2011 ]
 ]

 if Weather = "Rothamsted (2009)" [ set foragingHoursList foragingHoursListRothamsted2009 ]
 if Weather = "Rothamsted (2010)" [ set foragingHoursList foragingHoursListRothamsted2010 ]
 if Weather = "Rothamsted (2011)" [ set foragingHoursList foragingHoursListRothamsted2011 ]

 if Weather = "Berlin (2000-2006)"
 [
  let inputYear 2006 + round ((ceiling (ticks / 365)) mod 7)
  if inputYear > 2006 [ while [inputYear > 2006] [ set inputYear inputYear - 7] ]
    ; after 7 years, 1st dataset is used again etc.
  if inputYear = 2000 [ set foragingHoursList foragingHoursListBerlin2000 ]
  if inputYear = 2001 [ set foragingHoursList foragingHoursListBerlin2001 ]
  if inputYear = 2002 [ set foragingHoursList foragingHoursListBerlin2002 ]
  if inputYear = 2003 [ set foragingHoursList foragingHoursListBerlin2003 ]
  if inputYear = 2004 [ set foragingHoursList foragingHoursListBerlin2004 ]
  if inputYear = 2005 [ set foragingHoursList foragingHoursListBerlin2005 ]
  if inputYear = 2006 [ set foragingHoursList foragingHoursListBerlin2006 ]
 ]

 if Weather = "Berlin (2000)" [ set foragingHoursList foragingHoursListBerlin2000 ]

 if Weather != "HoPoMo_Season"
  and Weather != "HoPoMo_Season_Random"
  and Weather != "Constant"
  and Weather != "Weather File"    ;  ***NEW FOR BEEHAVE_BEEMAPP2015***
 [
  set foragingPeriod_s (item (day - 1) foragingHoursList) * 3600
 ] ; [s] hours sunshine on that day, in seconds

 if Weather = "HoPoMo_Season" or Weather = "HoPoMo_Season_Random"
 [
```

```
  set foragingPeriod_s 12 * 3600 * (1 - Season_HoPoMoREP day [ 385 25 36 155 60 ])
  if foragingPeriod_s < 3600 [ set foragingPeriod_s 0 ]
 ]   ; bell shape curve of foraging period, 12 * 3600 = 12 hrs max.

 if Weather = "HoPoMo_Season_Random"
 [
  if random-float 1 < 0.15 [ set foragingPeriod_s random-float (4 * 3600)]
 ]

 if Weather = "Constant" [ set foragingPeriod_s 8 * 3600 ]


 ;  begin ***NEW FOR BEEHAVE_BEEMAPP2015***
 if Weather = "Weather File"
 [
   let year_no ceiling (ticks / 365) - 1

   set foragingPeriod_s item (day - 1) ( item ( year_no mod length(WeatherDataList))
WeatherDataList) * 3600
 ]
 ; end ;  ***NEW FOR BEEHAVE_BEEMAPP2015***


 ask signs with [ shape = "sun"]
 [
  ifelse foragingPeriod_s > 0
    [ show-turtle set label precision (foragingPeriod_s / 3600) 1 ]
    [ hide-turtle set label " " ]
 ]   ; "sun" sign is shown, whenever there is an opportunity to forage

 ask signs with [ shape = "cloud"]
 [
  ifelse foragingPeriod_s < (4 * 3600)
    [ show-turtle ]
    [ hide-turtle ]
 ]   ; "cloud" sign is shown, whenever there is less than 4 hrs of foraging possible

 if foragingPeriod_s = -1
 [
  set BugAlarm true
  show "BugAlarm in Foraging_PeriodREP! Weather not defined!"
 ]
 if foragingPeriod_s < 0 [ set foragingPeriod_s 0 ]  ;  ***NEW FOR BEEHAVE_BEEMAPP2015***
 report foragingPeriod_s
end

;
**********************************************************************************
**********************************************************************************
************
```

```
to-report Foraging_ProbabilityREP
 ; calculates the probability that a forager start spontaneously to forage,
 ; called by Start_IBM_Proc once a day
 let foragingProbability 0.01  ; 0.01
  ; default foraging probability per "round" (round: ca. 13 min)
  ; 0.01 comparable to Dornhaus et al 2006: 0.00033/36s
 let highForProb 0.05 ; 0.02
 let emergencyProb 0.2
  ; foraging prob. is increased if pollen is needed:
 if (PollenStore_g / IdealPollenStore_g) < 0.2
 [
  set foragingProbability highForProb
 ]

 if HoneyEnergyStore / DecentHoneyEnergyStore < 0.5
 [
  set foragingProbability highForProb
 ]
 ; foraging prob. is increased if pollen is needed:
 if HoneyEnergyStore / DecentHoneyEnergyStore < 0.2
 [
  set foragingProbability emergencyProb
 ]

 if (PollenStore_g / IdealPollenStore_g) > 0.5 and
   HoneyEnergyStore / DecentHoneyEnergyStore > 1
   [
    set foragingProbability 0
   ] ; no foraging if plenty of honey and pollen is present

 let i 1
  while [ i <= N_GENERIC_PLOTS ]
  [
   let plotname (word "Generic plot " i)
    ; e.g. "Generic plot 1"
   set-current-plot plotname
   if (i = 1 and GenericPlot1 = "foraging probability")
   or (i = 2 and GenericPlot2 = "foraging probability")
   or (i = 3 and GenericPlot3 = "foraging probability")
   or (i = 4 and GenericPlot4 = "foraging probability")
   or (i = 5 and GenericPlot5 = "foraging probability")
   or (i = 6 and GenericPlot6 = "foraging probability")
   or (i = 7 and GenericPlot7 = "foraging probability")
   or (i = 8 and GenericPlot8 = "foraging probability")
   [
    create-temporary-plot-pen "ForProb"
    set-plot-pen-mode 0 ; 0: lines
    plotxy ticks  (foragingProbability)
   ]
    set i i + 1
```

```
  ]


 ask Signs with [shape = "exclamation"]
   [ ; if the foraging prob. is set to 0, an exclamation mark is shown
     ; on the interface (beside the weather sign)
     ifelse foragingProbability > 0
       [ hide-turtle ]
       [ show-turtle ]
   ]

 report foragingProbability
end

;
**********************************************************************************
**********************************************************************************
************

to Foraging_start-stopProc
  ; decision for pollen or nectar foraging; active foragers may quit foraging;
  ; foragers might spontaneously start or continue foraging (either exploiting known
  ; patch or search new patch)

  let FORAGE_AUTOCORR 0 ;
  ; autocorrelation of chosen forage (i.e. probability to not-reconsider chosen forage
  ; type: 1: always collect the same forage type (i.e. nectar!) if 0: no effect)

  ask foragerSquadrons with [ activity != "recForaging" ]
   ; this does not apply to bees, that followed a dance in the last foraging round
   ; and hence have already made their decision for nectar or pollen foraging
  [
    if random-float 1 > FORAGE_AUTOCORR
    ; if smaller, the bee sticks to her current food type
    [
      ifelse random-float 1 < ProbPollenCollection
        [
          set pollenForager true     ; IF -> pollen forager
          set activityList lput "PF" activityList
        ]
        [
          set pollenForager false    ; ELSE -> nectar forager
          set shape "bee_mb_1" ; ] ] ]
          set activityList lput "NF" activityList
        ]
    ]
  ]

  ask foragerSquadrons with
    [ activity != "resting"
      and activity != "recForaging"
```

```
    and activity != "lazy" ]
  ; i.e. ask actively foraging bees
[
 if random-float 1 < FORAGING_STOP_PROB
 ; active foragers, that weren't recruited in the foraging round before, may abandon foraging
  [
   set activity "resting"
   set activityList lput "AfR" activityList
  ]
]

; recording of the activities & forage type in the activityList
ask foragerSquadrons with
  [ activity = "searching" ]
  [
   if pollenForager = true
   [
    set activityList lput "Sp" activityList
   ]
   if pollenForager = false
   [
    set activityList lput "Sn" activityList
   ]
                                        ]
ask foragerSquadrons with [ activity = "resting" ]
[
 set activityList lput "R" activityList
]

ask foragerSquadrons with [ activity = "lazy" ]
[
 set activityList lput "L" activityList
]

ask foragerSquadrons with
 [ knownNectarPatch >= 0
  and pollenForager = false
 ]
 ; ask experienced NECTAR foragers if they abandon their nectar patch
[
 if random-float 1 < 1 / [ EEF ] of flowerPatch knownNectarPatch
  and random-float 1 < (HoneyEnergyStore / DecentHoneyEnergyStore)
   ; chance to abandon depends on 1/EEF and is reduced if colony needs nectar
 [
  set knownNectarPatch -1   ; forager doesn't know a nectar patch anymore
  ifelse (activity != "resting" and activity != "lazy")
   [
    set activity "searching"
    set activityList lput "AnSn" activityList
   ] ; active foragers that abandoned their patch have to search a new one
   [
```

```
        set activityList lput "An" activityList
      ]  ; resting foragers that abandoned their patch still rest
   ]
  ]

  ask foragerSquadrons with
   [ knownPollenPatch >= 0
     and pollenForager = true ]
   ; ask experienced POLLEN foragers if they abandon their pollen patch
  [
   if random-float 1 < 1 - (1 -
      ABANDON_POLLEN_PATCH_PROB_PER_S) ^ [ tripDurationPollen ] of flowerPatch
knownPollenPatch
    [
     set knownPollenPatch -1   ; forager doesn't know a pollen patch anymore
     ifelse ( activity != "resting"
       and activity != "lazy")
       [
         set activity "searching"
         set activityList lput "ApSp" activityList
       ] ; active foragers that abandoned their patch have to search a new one
       [
         set activityList lput "Ap" activityList
       ]  ; resting foragers that abandoned their patch still rest
   ]
  ]

  ask foragerSquadrons with [ activity = "resting" ]
  [
   if random-float 1 < ForagingSpontaneousProb
     ;  resting foragers may start foraging spontaneously..
   [
     if pollenForager = false
       ; ask (resting) nectar foragers to become active
     [
       ifelse knownNectarPatch >= 0
         [
           set activity "expForaging"
           set activityList lput "Xn" activityList
         ] ; IF they already know a NECTAR patch, they become experienced nectar foragers
         [
           set activity "searching"
           set activityList lput "Sn" activityList
         ] ; ELSE: they become scouts and search a new one
     ]

     if pollenForager = true   ; ask (resting) pollen foragers to become active
     [
       ifelse knownPollenPatch >= 0
         [
           set activity "expForaging"
```

```
        set activityList lput "Xp1" activityList
       ] ; IF they already know a POLLEN patch, they become experienced pollen foragers
       [
        set activity "searching"
        set activityList lput "Sp" activityList
       ] ; ELSE: they become scouts and search a new one
    ]
  ] ; "if random-float 1 < ForagingSpontaneousProb"
 ]
 ask foragerSquadrons  ; if bees are "exhausted" they cease foraging on that day:
 [
  if km_today >= MAX_km_PER_DAY
  [
   set activity "resting"
  ]
 ]

end

;
**********************************************************************************
**********************************************************************************
************

to Foraging_searchingProc
 ; called by: ForagingRoundProc, determines if a patch (and which one) is
 ; found by a searching forager

 let patchCounter 0
 let probSum 0  ; necessary to decide, which flower patch is found
 let chosenPatch -1  ; -1: i.e. no patch chosen yet
 let cumulative_NON-detectionProb 1
 let nowAvailablePatchesList [ ]

 ask flowerPatches with
   [ quantityMyl >= CROPVOLUME * SQUADRON_SIZE
    or amountPollen_g >= POLLENLOAD * SQUADRON_SIZE ]
   ; only patches with enough nectar OR pollen left are considered
 [
  set probSum probSum + detectionProbability ; sums up the detection probabilities of patches, to
decide later, which patch was actually found
  set cumulative_NON-detectionProb
   cumulative_NON-detectionProb * (1 - detectionProbability)
    ; Probability to find any patch is: 1 - Probability, to find no patch at all
  set nowAvailablePatchesList fput who nowAvailablePatchesList
 ]

 set TotalFPdetectionProb (1 - cumulative_NON-detectionProb)
  ; Probability to find ANY (not empty!) flower patch during one search trip

 ask foragerSquadrons with [ activity = "searching" ]
```

```
[
 set SearchingFlightsToday SearchingFlightsToday + SQUADRON_SIZE
  ; counts the numer of search flights on current day
 ifelse random-float 1 < TotalFPdetectionProb
  ; if any (not empty!!) flower patch found by the forager:
  [
   let p random-float probSum   ; to decide which flower patch is found
   set patchCounter 0
   set chosenPatch -1

   foreach nowAvailablePatchesList
   [
    ask flowerPatch ?   ;  "?" item of the list
    [  ; the patch is randomly chosen, according to its detection probability:
     set patchCounter patchCounter + detectionProbability
     if (patchCounter >= p) and (chosenPatch = -1) [ set chosenPatch who ]
    ]
   ]

   ifelse pollenForager = false
    [ set knownNectarPatch chosenPatch ]
     ; IF nectar forager: detected patch is memorised as nectar patch
    [ set knownPollenPatch chosenPatch ]
     ; ELSE pollen forager: detected patch is memorised as pollen patch

   if (knownNectarPatch < 0 and knownPollenPatch < 0)
   [
    user-message "BUG: negative flower patches!"
    set BugAlarm true
   ]

   ifelse ( pollenForager = false
    and [ quantityMyl ] of flowerPatch chosenPatch >= (CROPVOLUME * SQUADRON_SIZE))
     ; collection of NECTAR - only if nectar is available at the chosen patch!
     ; this is necessary as the patch may offer only pollen
    [
     set activity "bringingNectar"  ; then the scout becomes a successful nectar forager
     set activityList lput "fN" activityList

     ask flowerPatch knownNectarPatch
     [
      set quantityMyl (quantityMyl - (CROPVOLUME * SQUADRON_SIZE))
       ; quantity of nectar in patch is reduced

      set nectarVisitsToday nectarVisitsToday + SQUADRON_SIZE
      set summedVisitors summedVisitors + SQUADRON_SIZE
     ] ; and numbers of visitors increased
    ]
    [ ; ELSE: found a patch but it doesn't offer nectar: feN: "found empty nectar patch"
     if pollenForager = false
     [
```

```
          set knownNectarPatch -1
          set activityList lput "feN" activityList
         ]
        ]

      ifelse ( pollenForager = true
       and [ amountPollen_g ] of flowerPatch chosenPatch >= (POLLENLOAD * SQUADRON_SIZE))
        ; collection of POLLEN - only if pollen is available at the chosen patch!
       [
        set activity "bringingPollen"  ; then the scout becomes a successful pollen forager
        set activityList lput "fP" activityList

        ask flowerPatch knownPollenPatch
        [
         set amountPollen_g (amountPollen_g - (POLLENLOAD * SQUADRON_SIZE))
          ; quantity of nectar in patch is reduced
         set pollenVisitsToday pollenVisitsToday + SQUADRON_SIZE
         set summedVisitors summedVisitors + SQUADRON_SIZE
        ] ; and numbers of visitors increased
       ]
       [
        if pollenForager = true
        [
         set knownPollenPatch -1
         set activityList lput "feP" activityList
        ]
       ] ; ELSE: found patch does not offer nectar: feP: "found empty pollen patch"
   ] ; "ifelse random-float 1 < TotalFPdetectionProb"

   ; ELSE: no patch is found; uS = unsuccessful searching
   [
    set activityList lput "uS" activityList
   ]
] ; "ask foragerSquadrons with [ activity = "searching" ]"

ask foragerSquadrons with ; ask recruited NECTAR foragers:
  [ activity = "recForaging" ; forager is recruited
    and knownNectarPatch >= 0 ; it knows a patch where it is recruited to
    and pollenForager = false ] ; and it is looking for nectar
[ ; the flights of recruited bees are counted:
 set RecruitedFlightsToday RecruitedFlightsToday + SQUADRON_SIZE
 ; IF(1) recruited Forager finds the nectar patch:
 ifelse random-float 1 < FIND_DANCED_PATCH_PROB
  [ ; and IF (2) nectar is still there:
    ifelse [ quantityMyl ] of flowerPatch knownNectarPatch >= (CROPVOLUME * SQUADRON_SIZE)
     [  ; .. then the recruit becomes a successful nectar forager
       set activity "bringingNectar"
       ; which is recorded in its activityList:
       set activityList lput "frN" activityList
       ask flowerPatch knownNectarPatch
       [  ; the nectar in the patch is then reduced:
```

```
          set quantityMyl (quantityMyl - (CROPVOLUME * SQUADRON_SIZE))
          ; the visit is counted:
          set nectarVisitsToday nectarVisitsToday + SQUADRON_SIZE
          set summedVisitors summedVisitors + SQUADRON_SIZE
         ]
        ]
        [  ; ELSE(2): if patch has not enough nectar, recruit becomes a scout again
         set activity "searching"
         set activityList lput "eSn" activityList
         ; and the patch is forgotten:
         set knownNectarPatch -1
        ]
    ]
    [ ; ELSE(1): if the recruits does not find the patch, it starts searching
     set activity "searching"
     set activityList lput "mSn" activityList
     ; and forgets "known" nectar patch
     set knownNectarPatch -1
    ]
 ]

 ; also recruited POLLEN foragers are searching a patch:
 ask foragerSquadrons with
   [ activity = "recForaging"
    and knownPollenPatch >= 0
    and pollenForager = true ]
 [
  set RecruitedFlightsToday RecruitedFlightsToday + SQUADRON_SIZE
  ; they find their patch with the probability of FIND_DANCED_PATCH_PROB
  ifelse random-float 1 < FIND_DANCED_PATCH_PROB
   ; IF(1) recruited Forager finds the pollen patch...
   [
    ifelse [ amountPollen_g ] of flowerPatch knownPollenPatch >= (POLLENLOAD *
SQUADRON_SIZE)
      ; ..and pollen is still there..
      [ set activity "bringingPollen"
       ; .. then the recruit becomes a successful pollen forager
       set activityList lput "frP" activityList
       ask flowerPatch knownPollenPatch
       [
        set amountPollen_g (amountPollen_g - (POLLENLOAD * SQUADRON_SIZE))
         ; ..pollen in the patch is reduced
        set pollenVisitsToday pollenVisitsToday + SQUADRON_SIZE
        set summedVisitors summedVisitors + SQUADRON_SIZE
       ]  ; ..and numbers of visitors increased
      ]
      [ ; ELSE(2): if patch has not enough pollen, recruit becomes a scout again
       set activity "searching"
       set activityList lput "eSp" activityList
       set knownPollenPatch -1
      ]
```

```
    ]
    [ ; ELSE(1): if she does not find the patch, she starts searching
      ; (but can't find another patch in this foraging round)
      set activity "searching"
      set activityList lput "mSp" activityList
      ; it forgets its "known" pollen patch:
      set knownPollenPatch -1
    ]
 ] ; "ask foragerSquadrons with [ activity = "recForaging"]"

end;

;
**********************************************************************************
**********************************************************************************
************

to Foraging_collectNectarPollenProc
 ; successful foragers gather nectar/pollen (if still available) and decrease
 ; nectar/pollen in flower patch

  ; ask experienced NECTAR foragers:
 ask foragerSquadrons with
   [ activity = "expForaging"
     and knownNectarPatch >= 0
     and pollenForager = false ]
 [ ; does patch still have enough nectar?:
  ifelse [ quantityMyl ] of flowerPatch knownNectarPatch >= (CROPVOLUME * SQUADRON_SIZE)
    [ ; the forager will then be bringing nectar:
      set NectarFlightsToday NectarFlightsToday + SQUADRON_SIZE
      set activity "bringingNectar"
      ; this is recorded in its activityList:
      set activityList lput "N" activityList

      ask flowerPatch knownNectarPatch
      [ ; available nectar in the patch is reduced:
        set quantityMyl (quantityMyl - ( CROPVOLUME * SQUADRON_SIZE))
        ; the visits are counted:
        set nectarVisitsToday nectarVisitsToday + SQUADRON_SIZE
        ; and numbers of visitors increased:
        set summedVisitors summedVisitors + SQUADRON_SIZE
      ]
    ]
    [ ; ELSE: not enough nectar available at the patch
      ; the forager will then become a scout:
      set activity "searching"
      set activityList lput "eSn" activityList
      ; the bee forgets this empty nectar patch
      set knownNectarPatch -1
    ]
 ]
```

```
; ask experienced POLLEN foragers:
ask foragerSquadrons with
  [ activity = "expForaging"
    and knownPollenPatch >= 0
    and pollenForager = true ]
[ ; does patch still have enough pollen?
 ifelse [ amountPollen_g ] of flowerPatch knownPollenPatch >= (POLLENLOAD * SQUADRON_SIZE)
  [ ; IF patch has enough pollen:
    set PollenFlightsToday PollenFlightsToday + SQUADRON_SIZE
    ; the forager will then be bringing pollen:
    set activity "bringingPollen"
    set activityList lput "P" activityList

    ask flowerPatch knownPollenPatch
    [  ; available pollen in the patch is reduced:
      set amountPollen_g (amountPollen_g - (POLLENLOAD * SQUADRON_SIZE))
      set pollenVisitsToday pollenVisitsToday + SQUADRON_SIZE
       ; and numbers of visitors increased
      set summedVisitors summedVisitors + SQUADRON_SIZE ]
  ]
  [ ; ELSE: not enough pollen available at the patch
    ; the forager will then become a scout:
    set activity "searching"
    set activityList lput "eSp" activityList
    set knownPollenPatch -1
  ]
]

; experienced pollen foragers, who know a nectar patch but no pollen patch
; or experienced nectar foragers, who know a pollen patch but no nectar patch:
; this can happen if e.g. an exp. nectar foragers switches to pollen foraging
; these bees switch to "resting" and DO NOT LEAVE THE HIVE!
; hence, their mileometer or km_today doesn't change
; and they are not considered in the Foraging_MortalityProc
ask foragerSquadrons with
  [ ( activity = "expForaging"    ; experienced (but got its experience as pollen forager!)
      and pollenForager = false   ; has now switched to nectar foraging
      and knownNectarPatch = -1   ; but doesn't know a nectar patch
      )
      or
    ( activity = "expForaging"    ; experienced (but got its experience as nectar forager!)
      and pollenForager = true    ; has now switched to pollen foraging
      and knownPollenPatch = -1   ; but doesn't know a pollen patch
      )]
[
  set activity "resting"      ; switch to resting - i.e. they haven't left the hive in this foraging round
  set activityList lput "Rx" activityList
]

; ask successful NECTAR foragers:
```

```
ask foragerSquadrons with [ activity = "bringingNectar" ]
[ ; the energy content of their cropload is calculated, which depends on the nectar concentration:
  set cropEnergyLoad ([ nectarConcFlowerPatch ] of
    flowerPatch knownNectarPatch * CROPVOLUME * ENERGY_SUCROSE)  ; [kJ]

  ; the distance they have travelled today is increased..
  set km_today km_today + ([ flightCostsNectar ] of
    flowerPatch knownNectarPatch / (FLIGHTCOSTS_PER_m * 1000))

  ; and also their total travelled distance:
  set mileometer mileometer + ([ flightCostsNectar ] of
   flowerPatch knownNectarPatch / (FLIGHTCOSTS_PER_m * 1000))  ;

  ifelse readInfile = true
   [ ; if patch data are read in, then the color of the bee
     ; reflects the ID of the flower patch:
     ; set color knownNectarPatch
     let memoColor 0
     ask flowerPatch knownNectarPatch [ set memoColor color ]
     set color memoColor
   ]
   [ ; ELSE: if there are 2 patches, defined via GUI,
     ; then the color of the bee reflects the patch it is foraging at:
     if knownNectarPatch = -1 [ set color grey ]
     if knownNectarPatch = 0 [ set color red ]
     if knownNectarPatch > 0 [ set color green ]
   ]
]

; and similar for successful POLLEN foragers:
ask foragerSquadrons with [ activity = "bringingPollen" ]
[ ; the pollen load is the same for all patches!
  set collectedPollen POLLENLOAD ; [g]
  set shape "bee_mb_pollen"

  ; the distance they have travelled today is increased..
  set km_today km_today + ([ flightCostsPollen ] of
   flowerPatch knownPollenPatch / (FLIGHTCOSTS_PER_m * 1000))

  ; and also their total travelled distance:
  set mileometer mileometer + ([ flightCostsPollen ] of
   flowerPatch knownPollenPatch / (FLIGHTCOSTS_PER_m * 1000))  ;

  ifelse readInfile = true
   [ ; the color of the bee is set according to its flower patch:
     ; set color knownPollenPatch
     let memoColor 0
     ask flowerPatch knownPollenPatch [ set memoColor color ]
     set color memoColor
   ]
   [
```

```
      if knownPollenPatch = -1 [ set color grey ]
      if knownPollenPatch = 0 [ set color red ]
      if knownPollenPatch > 0 [ set color green ]
    ]
  ]
end;

;
**********************************************************************************
**********************************************************************************
************

to Foraging_flightCosts_flightTimeProc
  ; sums up travelled distance for unsuccessful scouts and honey consumption due to foraging, trip
duration
  ; consumption is subtracted from honey store, not from the crop, as it is empty for unsuccessful
scouts
  let energyConsumption 0

  ; flight distance for successful foragers is calculated in Foraging_collectNectarPollenProc!
  ; flight distance for unsuccessful scout is calculated here:
  ask foragerSquadrons with [ activity = "searching" ]
  [ ; the search length [m] of the foraging trip is added to today's km and the lifetime km
(mileometer):
    set km_today km_today + ( SEARCH_LENGTH_M / 1000 )
    set mileometer mileometer + ( SEARCH_LENGTH_M / 1000 )  ; mileometer: [km]

    ; honey store in the colony is reduced to reflect the energy consumed during the trip:
    set HoneyEnergyStore HoneyEnergyStore - ( SEARCH_LENGTH_M * FLIGHTCOSTS_PER_m *
SQUADRON_SIZE )
    set ColonyTripDurationSum ColonyTripDurationSum + (SEARCH_LENGTH_M / FLIGHT_VELOCITY )
; sums up time of a search trip

    ; sums up # foragers doing a trip & unsuccessful foraging trips:
    set ColonyTripForagersSum ColonyTripForagersSum + 1
    set EmptyFlightsToday EmptyFlightsToday + SQUADRON_SIZE
  ]

  ; energy consumption for successful foragers:
  ask foragerSquadrons with
    [ activity = "bringingNectar"
      or activity = "bringingPollen" ]
  [
    if pollenForager = false ; ask NECTAR foragers
    [
      ask flowerPatch knownNectarPatch
      [ ; flightCostsNectar is a flowerPatch variable, reflecting distance and handling time
        set energyConsumption flightCostsNectar
        ; energy is used, according to the flight costs of the patch
        set ColonyTripDurationSum ColonyTripDurationSum + tripDuration + TIME_UNLOADING
```

```
    ]  ; adds duration of this nectar trip to the sum of all trips performed during this foraging round so
far
   ]
   if pollenForager = true ; ask POLLEN foragers
    [
     ask flowerPatch knownPollenPatch
     [
      set energyConsumption flightCostsPollen
       ; energy is used, according to the flight costs of the patch
      set ColonyTripDurationSum ColonyTripDurationSum + tripDurationPollen +
TIME_UNLOADING_POLLEN
     ] ; adds duration of this pollen trip to the sum of all trips performed during this foraging round
so far
    ]

    ; colony's honey store is decreased:
   set HoneyEnergyStore HoneyEnergyStore - ( energyConsumption * SQUADRON_SIZE)
    ; sums up # foragers doing a trip:
   set ColonyTripForagersSum ColonyTripForagersSum + 1
 ]

end


;
**********************************************************************************
**********************************************************************************
************

to Foraging_mortalityProc
 ; mortality of foragers during their foraging trip, counts # dying foragers and their lifespan
 let emptyTripDuration SEARCH_LENGTH_M / FLIGHT_VELOCITY  ; [s] = 10 min

 ask foragerSquadrons with [ activity = "searching" ]
 [ ; mortality risk of unsuccessful scouts depends on their time spent for searching
  ; mortality risk calculated as probability to NOT survive every single second of the foraging trip:
  if random-float 1 < 1 - ((1 - MORTALITY_FOR_PER_SEC) ^ emptyTripDuration)
  [ ; deaths are counted and the lifespans summed up to later calculate a mean lifespan:
   set DeathsAdultWorkers_t DeathsAdultWorkers_t + SQUADRON_SIZE
   set DeathsForagingToday DeathsForagingToday + SQUADRON_SIZE
   set SumLifeSpanAdultWorkers_t SumLifeSpanAdultWorkers_t + (age * SQUADRON_SIZE)
   die
  ]
 ]
 ; this is similar for NECTAR foragers, but here with a patch specific mortalityRisk
 ask foragerSquadrons with [ activity = "bringingNectar" ]
 [
  if random-float 1 < ([ mortalityRisk ] of flowerPatch knownNectarPatch)
  [
   set DeathsAdultWorkers_t DeathsAdultWorkers_t + SQUADRON_SIZE
   set DeathsForagingToday DeathsForagingToday + SQUADRON_SIZE
```

```
    set SumLifeSpanAdultWorkers_t SumLifeSpanAdultWorkers_t + (age * SQUADRON_SIZE)
    die
   ]
  ]
  ; and again for POLLEN foragers, with a patch specific mortalityRiskPollen:
  ask foragerSquadrons with [ activity = "bringingPollen" ]
  [
   if random-float 1 < ([ mortalityRiskPollen ] of flowerPatch knownPollenPatch)
   [
    set DeathsAdultWorkers_t DeathsAdultWorkers_t + SQUADRON_SIZE
    set DeathsForagingToday DeathsForagingToday + SQUADRON_SIZE
    set SumLifeSpanAdultWorkers_t SumLifeSpanAdultWorkers_t + (age * SQUADRON_SIZE)
    die
   ]
  ]
end;

;
*******************************************************************************
*******************************************************************************
************

to Foraging_dancingProc
  ; foragers dance for a good patch and recruit 2 pollen foragers or up to 5 nectar foragers
  ; to the advertised patch

  let EEFdancedPatch -999   ; correct number set later
   ; energetic efficiency of the flower patch danced for (set to nonsense number as control)

  let tripDurationDancedPatch -999  ; correct number set later
   ; trip duration to a pollen patch

  let patchNumberDanced -999  ; correct number set later
   ; ...and the number of that flower patch

  ask foragerSquadrons with
   [ activity = "bringingNectar"
    or activity = "bringingPollen" ]
    ; successful pollen or nectar foragers are addressed
  [
   if activity = "bringingNectar" ; NECTAR FORAGERS
   [
    set EEFdancedPatch [ EEF ] of flowerPatch knownNectarPatch
    set patchNumberDanced knownNectarPatch
     ; successful foragers dance; they communicate EEF and ID of flowerPatch

    let danceFollowersNectarNow
     random-poisson [ danceFollowersNectar ] of flowerPatch knownNectarPatch

    if [ danceFollowersNectarNow ] of flowerPatch knownNectarPatch >= 1
    [
```

```
    set activityList lput "Dn" activityList
   ]

   if ( count foragerSquadrons with
      [ activity = "resting" ]) >=
         [ danceFollowersNectarNow ] of flowerPatch knownNectarPatch
         ; only if enough resting foragers are present, there will be dances
   [
    ask n-of
         ([ danceFollowersNectarNow ] of flowerPatch knownNectarPatch)
           foragerSquadrons with [ activity = "resting" ]
             ; depending on EEF of the patch, (0-5) resting foragers will follow the dance
    [
     ifelse knownNectarPatch = -1
      [    ; unexperienced foragers will always accept the advertised patch:
       set knownNectarPatch patchNumberDanced
       set activity "recForaging"
       set pollenForager false
        ; and become a nectar forager
       set activityList lput "rFnNF" activityList
      ]
      [
       ifelse EEFdancedPatch > [ EEF ] of flowerPatch knownNectarPatch
        ; if(2) ; experienced foragers: if the advertised patch has higher EEF
        ; than the known flowerPatch,
        [
         set knownNectarPatch patchNumberDanced
          ; the dance follower will switch to new patch

         set pollenForager false
          ; and become a nectar forager

         set activity "recForaging"
         set activityList lput "rFnxNF" activityList
        ]
        [ ; ELSE 2 (i.e. experienced foragers, knowing a BETTER patch) are activated
         set activity "expForaging"
         set activityList lput "Xnr" activityList
        ]  ; else (2) they become active foragers to their own, known patch
      ]
    ]
   ]
  ]

  if activity = "bringingPollen"                                    ; POLLEN FORAGERS
  [
   set tripDurationDancedPatch [ tripDurationPollen ] of flowerPatch knownPollenPatch
   set patchNumberDanced knownPollenPatch
   if POLLEN_DANCE_FOLLOWERS >= 1 ; pollen foragers dance ALWAYS (as
POLLEN_DANCE_FOLLOWERS = 2)
   [
```

```
      set activityList lput "Dp" activityList
    ]

    if ( count foragerSquadrons with [ activity = "resting" ])
     >= POLLEN_DANCE_FOLLOWERS
       ; only if enough resting foragers are present, there will be dances
    [
     ask n-of POLLEN_DANCE_FOLLOWERS foragerSquadrons
      with [ activity = "resting" ]
      ; # pollen dance followers: constant and independent of patch distance!!
     [
      ifelse knownPollenPatch = -1
       [  ; unexperienced forager will always accept the advertised patch:
         set knownPollenPatch patchNumberDanced
         set activity "recForaging"

         ; and become a pollen forager:
         set pollenForager true
         set activityList lput "rFpPF" activityList
       ]
       [ ; if(2) ; experienced foragers: if the advertised patch offers a
         ;  shorter trip duration than the known pollen patch..
         ifelse tripDurationDancedPatch < [ tripDurationPollen ]
          of flowerPatch knownPollenPatch
          [ ; .. then the dance follower will switch to new patch
           set knownPollenPatch patchNumberDanced
           ; and become a pollen forager:
           set pollenForager true

           set activity "recForaging"
           set activityList lput "rFpxPF" activityList
          ]
          [ ; else (2) they become active foragers to their own, known patch:
           set activity "expForaging"
           set activityList lput "Xpr" activityList
          ]
       ]
     ]
    ]
   ]
  ]

end;

;
*******************************************************************************
*******************************************************************************
************

to Foraging_unloadingProc
  ; successful foragers increase honey or pollen store of the colony and become experienced foragers
```

```
 ask foragerSquadrons with [ activity = "bringingNectar" ]
 [
  set HoneyEnergyStore HoneyEnergyStore + (cropEnergyLoad * SQUADRON_SIZE)

  if HoneyEnergyStore > MAX_HONEY_ENERGY_STORE
  [
   set HoneyEnergyStore MAX_HONEY_ENERGY_STORE
  ]  ; honey store can't be larger than maximum

  set activityList lput "bN" activityList
  set cropEnergyLoad 0
  set activity "expForaging"
  set activityList lput "Xn" activityList
 ]

 ask foragerSquadrons with [ activity = "bringingPollen" ]
 [
  set PollenStore_g PollenStore_g + (collectedPollen * SQUADRON_SIZE)
  set collectedPollen 0
  set activityList lput "bP" activityList
  set activity "expForaging"
  set activityList lput "Xp" activityList
 ]

 ask foragerSquadrons with [ activity = "searching" ]
 [
  set activityList lput "E" activityList
 ]  ; unsuccessful souts return empty

end;

;
**********************************************************************************
**********************************************************************************
************

to ForagersLifespanProc
 ; foragers also die due to age, max. travelled distance or by chance inside
 ; the colony; dying foragers are counted to calculate mean lifespan

 ask foragerSquadrons
 [
  if age >= LIFESPAN
  [
   set DeathsAdultWorkers_t DeathsAdultWorkers_t + SQUADRON_SIZE
   set SumLifeSpanAdultWorkers_t SumLifeSpanAdultWorkers_t + (age * SQUADRON_SIZE)
   die
  ]

  if mileometer >= MAX_TOTAL_KM
```

```
    [
      set DeathsAdultWorkers_t DeathsAdultWorkers_t + SQUADRON_SIZE
      set DeathsForagingToday DeathsForagingToday + SQUADRON_SIZE
      set SumLifeSpanAdultWorkers_t SumLifeSpanAdultWorkers_t + (age * SQUADRON_SIZE)
      die
    ]

    let dailyRiskToDie MORTALITY_INHIVE
      ; the daily background mortality of (healthy) foragers, which is equal to MORTALITY_INHIVE of
the inhive bees

    if infectionState = "infectedAsPupa"
    [
      set dailyRiskToDie MORTALITY_INHIVE_INFECTED_AS_PUPA
    ]  ; except for infected as pupa foragers, which have a higher mortality

    if infectionState = "infectedAsAdult"
    [
      set dailyRiskToDie MORTALITY_INHIVE_INFECTED_AS_ADULT
    ]  ; except for infected as adult foragers, which have a higher mortality

    if random-float 1 < dailyRiskToDie
    [
      set DeathsAdultWorkers_t DeathsAdultWorkers_t + SQUADRON_SIZE
      set SumLifeSpanAdultWorkers_t SumLifeSpanAdultWorkers_t + (age * SQUADRON_SIZE)
      die
    ]
  ] ; ask foragerSquadrons
end;

;
**********************************************************************************
**********************************************************************************
************

;
==========================================================================
==========================================================================
==========
;  ===============  END OF IBM FORAGING SUBMODEL
=============================================================  END OF IBM
FORAGING SUBMODEL ==========================
;
==========================================================================
==========================================================================
==========

;
**********************************************************************************
**********************************************************************************
************
```

```
;
*********************************************************************************************************************************
; ..............  THE VARROA MITE SUBMODEL  ..........................................  THE VARROA MITE SUBMODEL  ..........
;
*********************************************************************************************************************************

to MiteProc ; calls the Varroa related procedures
  CreateMiteOrganisersProc
  CountingProc ; updating number of brood & adults of drones & workers
  MitesInvasionProc
  MitePhoreticPhaseProc
  MiteDailyMortalityProc
  MiteOrganisersUpdateProc
end


;
*********************************************************************************************************************************

to CreateMiteOrganisersProc
  ; called by MiteProc, creates a single miteOrganiser turtle, that
  ; stores info on number and distribution of mites newly invaded into the brood cells

  create-miteOrganisers 1
  [
    setxy -1 -7
    set heading 0
    set size 1.3
    set color 33.5
    set shape "VarroaMite03"   ;"Virus1" ;"VarroaMite03"
    set workerCellListCondensed n-values (MAX_INVADED_MITES_WORKERCELL + 1) [ 0 ]
      ; +1 as also the number of mite free cells is stored in this list

    set droneCellListCondensed n-values (MAX_INVADED_MITES_DRONECELL + 1) [ 0 ]
      ; +1 as also the number of mite free cells is stored in this list

    set label-color white
    set cohortInvadedMitesSum 0
      ; sum of all mites that invaded a worker or drone cell on the same Day

    set invadedMitesHealthyRate PhoreticMitesHealthyRate
      ; rate of healthy mites in this cohort of invading mites equals the rate of healthy
```

```
    ; phoretic mites on this day

   set age INVADING_WORKER_CELLS_AGE
    ; "age" refers to age of invaded brood. If age for invasion differs in
    ; worker and drone brood..

   if INVADING_DRONE_CELLS_AGE < INVADING_WORKER_CELLS_AGE
   [
    set age INVADING_DRONE_CELLS_AGE
   ] ; ..then age refers to the younger of both
  ]
end

;
**********************************************************************************
**********************************************************************************
***********

to MitesInvasionProc
  ; called by MiteProc  calculates the number of phoretic mites that
  ; enter worker and drone brood cells on this day based on: Calis et al. 1999, Martin 2001

  let factorDrones 6.49  ; (Boot et al. 1995, Martin 2001)
  let factorWorkers 0.56 ; (Boot et al. 1995, Martin 2001)
  let adultsWeight_g (TotalIHbees + TotalForagers) * WEIGHT_WORKER_g
   ; weight of all adult worker bees
  let invadingBroodCellProb 0
   ; probability for a phoretic mite to enter any suitable brood cell
  let invadingWorkerCellProb 0
   ; probaility to invade a worker cell (only if any cell was invaded)
  let suitableWorkerCells 0
  let suitableDroneCells 0
   ; number of worker and drone cells, that are suitable for mite invasion
  let rD 0
  let rW 0
   ; rD, rW: Rate of invasion into Drone cells and Worker cells (Boot et al. 1995)

  ask larvaeCohorts with [ age = INVADING_WORKER_CELLS_AGE ]
  [
   set suitableWorkerCells number
  ] ; (age = 8) mites enter worker larvae cells ~1d before capping (at 9d age) (Boot, Calis, Beetsma
1992)

  ask droneLarvaeCohorts with [ age = INVADING_DRONE_CELLS_AGE ]
  [
   set suitableDroneCells number
  ]  ; (age = 8) mites enter drone larvae cells ~ 2d before capping (at 10d age) (Boot, Calis, Beetsma
1992)

  if adultsWeight_g > 0
  [ ; invasion rates in worker and drone cells:
```

```
   set rW factorWorkers * (suitableWorkerCells / adultsWeight_g)  ; (Martin 1998, 2001; Calis et
al.1999)
   set rD factorDrones * (suitableDroneCells / adultsWeight_g)
  ]

  let exitingMites 0
   ; # mites, that theoretically should invade cells but leave it immediatly,
   ; because the cell is already invaded by the max. number of mites

  let workerCellListTemporary n-values suitableWorkerCells [ 0 ]
   ; two temporary lists of all suitable worker/drone cells, to store
   ; the number of mites in each cell..

  let droneCellListTemporary n-values suitableDroneCells [ 0 ]
   ; .. of which later the number of cells invaded by 0, 1, 2.. mites can be calculated

  let cell -1
   ;  stores randomly chosen cell, which is invaded by a mite in the below
   ; "repeat phoreticMites.." process. -1 will be changed to a random number >= 0

  set InvadingMitesWorkerCellsTheo 0
  set InvadingMitesDroneCellsTheo 0
  set invadingBroodCellProb (1 - (exp (-(rW + rD))))
   ; probability for a phoretic mite to enter a brood cell; similar to
   ; Martin 2001, however: we use probability instead of proportion

  if rW + rD > 0 ; if invasion takes place..
  [
   set invadingWorkerCellProb (rW / (rW + rD))
  ]

  ; based on the Boot/Martin/Calis rates of cell invasion, which are used as probabilities,
  ; it is calculated how many phoretic mites enter a brood cell, and whether it is
  ; a drone or a worker cell; each invading mite is then associated with a random brood
  ; cell number (WorkerCellsInvasionList), finally, the mites in each "brood cell" are
  ; counted and saved in the condensed nMitesInCellsList
  repeat PhoreticMites
  [
   if random-float 1 < invadingBroodCellProb
    ; mites have a chance to enter a brood cell
   [
    ifelse random-float 1 < invadingWorkerCellProb ; the brood cell might be a WORKER cell
     [
      set InvadingMitesWorkerCellsTheo InvadingMitesWorkerCellsTheo + 1
       ; mites entering worker cells are counted

      set cell random suitableWorkerCells
       ; randomly, one of the suitable WORKER cells is invaded by a mite

      set WorkerCellListTemporary replace-item cell WorkerCellListTemporary
       (item cell WorkerCellListTemporary + 1)
```

```
        ; this list contains all worker cells and the number of mites
        ; invading into each cell
      ]
      [
       ; ELSE: invasion into DRONE cell
       set InvadingMitesDroneCellsTheo InvadingMitesDroneCellsTheo + 1
       set cell random suitableDroneCells
        ; randomly, one of the suitable drone cells is invaded by a mite

       set DroneCellListTemporary replace-item cell DroneCellListTemporary
        (item cell DroneCellListTemporary + 1)
        ; this list contains all drone cells and the number of mites
        ; invading into each cell
      ]
  ]
]

; excess of invaded mites: # mites in each cells is restricted to MAX_INVADED_MITES:
let counter 0
foreach WorkerCellListTemporary
[
  ; (note: items are addressed in ordered way - NOT randomly)
  if ? > MAX_INVADED_MITES_WORKERCELL
  [
    set exitingMites exitingMites + (? - MAX_INVADED_MITES_WORKERCELL)
     ; if too many mites in cells: they leave the cell ("?": # of mites in the cell)

    set WorkerCellListTemporary replace-item
     counter WorkerCellListTemporary MAX_INVADED_MITES_WORKERCELL
      ; .. mites left in the cell = max. mites in worker cell
  ]

  set counter counter + 1
]
set InvadingMitesWorkerCellsReal InvadingMitesWorkerCellsTheo - exitingMites

; and the same for the drones..
set counter 0  ; resetting the counter

foreach DroneCellListTemporary
[
  if ? > MAX_INVADED_MITES_DRONECELL
  [
    set exitingMites exitingMites + (? - MAX_INVADED_MITES_DRONECELL)
     ; if too many mites in cells: they leave the cell ("?": # of mites in the cell)

    set DroneCellListTemporary replace-item counter
     DroneCellListTemporary MAX_INVADED_MITES_DRONECELL
     ; .. mites left in the cell = max. mites in drone cell
  ]
  set counter counter + 1
```

```
]

set InvadingMitesDroneCellsReal InvadingMitesDroneCellsTheo
  - exitingMites
  + (InvadingMitesWorkerCellsTheo - InvadingMitesWorkerCellsReal)
    ; mites invaded drone cells = mites theor. invading drone cells
    ; - mites exiting drone&worker cells
    ; + mites exiting worker cells (here: exitingMites: sum of worker&drone cell mites!)

set PhoreticMites PhoreticMites
  - InvadingMitesWorkerCellsTheo
  - InvadingMitesDroneCellsTheo
  + exitingMites
    ; # of phoretic mites left (=phor.mites - invading mites
    ; + mites immediately leaving cells and become phoretic again

if PhoreticMites < 0
[
  user-message "Error in MitesInvasionProc - negative number of phoretic Mites"
  set BugAlarm true
] ; assertion

let memory -1 ; -1: = no cohort invaded

ask miteOrganisers with [age = INVADING_WORKER_CELLS_AGE]
[
  foreach workerCellListTemporary
    ; checks the list that contains all worker brood cells for
    ; how many mites have entered..
  [
    set workerCellListCondensed replace-item ? workerCellListCondensed
      ((item ? workerCellListCondensed) + 1)
  ]  ; sums up the number of cells entered by 0, 1,2..n mites in the mitesOrganisers own list

  set cohortInvadedMitesSum cohortInvadedMitesSum + InvadingMitesWorkerCellsReal

  let whoMO who  ; stores the "who" of the current miteOrganiser
  ask larvaeCohorts with [age = INVADING_WORKER_CELLS_AGE]
  [
    set invadedByMiteOrganiserID whoMO
    set memory who
  ]
  set invadedWorkerCohortID memory
] ; "ask miteorganisers ..."

ask miteOrganisers with [age = INVADING_DRONE_CELLS_AGE]
[
  foreach droneCellListTemporary
    ; checks the list that contains all drone brood cells for
    ; how many mites have entered..
  [
```

```
    set droneCellListCondensed replace-item ? droneCellListCondensed
      ((item ? droneCellListCondensed) + 1)
   ] ; sums up the cell entered by 0, 1,2..n mites in the mitesOrganisers own list

   set cohortInvadedMitesSum cohortInvadedMitesSum + InvadingMitesDroneCellsReal
   set memory -1  ; -1: = no cohort invaded

   ask droneLarvaeCohorts with [age = INVADING_DRONE_CELLS_AGE]
   [
    set memory who
   ]
   set invadedDroneCohortID memory
   let whoMO who  ; stores the "who" of the current miteOrganiser

   ask droneLarvaeCohorts with [ age = INVADING_DRONE_CELLS_AGE ]
   [
    set invadedByMiteOrganiserID whoMO
   ]
  ] ; "ask miteOrganisers with ..."

  if (PhoreticMites + InvadingMitesWorkerCellsReal
    +  InvadingMitesDroneCellsReal) > 0 ; avoid div 0!
  [
   set PropNewToAllPhorMites NewReleasedMitesToday
    / ( PhoreticMites + InvadingMitesWorkerCellsReal + InvadingMitesDroneCellsReal)
  ] ; Proportion of new emerged phoretic mites (today) to all phoretic mites
   ; present (needed in the MitePhoreticPhaseProc to determine # of newly infected phoretic mites
etc)
end

;
*******************************************************************************
*******************************************************************************
************

to-report MiteDensityFactorREP [ ploidyMiteOrg mitesIndex ]
  ; reports the (single) density factor for a certain number of invaded mites
  ; depending on ploidy of bee brood and chosen reproduction model

  let dataList []

  if MiteReproductionModel = "Martin"
  [ ifelse ploidyMiteOrg = 2
    [ set dataList [ 0 1 0.91 0.86 0.60 ] ]
     ; workers  (list length: 5) [ 0 1 0.91 0.86 0.60 ]
     ; from Martin 1998, Tab. 4; first value (0) doesn't matter, as no
     ; mother mite invaded these cells

    [ set dataList [ 0 1 0.84 0.65 0.66 ] ]
  ]   ; drones (list length: 5)  [ 0 1 0.84 0.65 0.66 ] from Martin 1998, Tab. 4
```

```
 if MiteReproductionModel = "Fuchs&Langenbach"
 [
  ifelse ploidyMiteOrg = 2
    [ set dataList [ 0 1 0.96 0.93 0.89 0.86 0.82 0.79 0 ]]
      ; workers   (list length: 9) calculated from Fuchs&Langenbach 1989 Tab.III
    [ set dataList [ 0 1 0.93 0.86 0.80 0.73 0.66 0.59 0.52 0.45 0.39 0.32 0.25 0.18 0.11 0.05 0 ] ]
 ]  ; (list length: 17) calculated from Fuchs&Langenbach 1989 Tab.III

 if MiteReproductionModel = "No Mite Reproduction"  ; only for model testing
 [
  ifelse ploidyMiteOrg = 2
    [ set dataList [ 0 1 1 1 1 1 ] ] ; workers   (list length: 6)
    [ set dataList [ 0 1 1 1 1 1 ] ]
 ]  ; drones  (list length: 6)

 if MiteReproductionModel = "Martin+0"
  ; like Martin, but max # of mites in brood cell is increased by
  ; one with a rel. reprod. rate of 0 (= 0 at the end of the list)

 [ ; Martin Test with 0
  ifelse ploidyMiteOrg = 2
    [ set dataList [ 0 1 0.91 0.86 0.60 0 ] ]
      ; workers   (list length: 6) [ 0 1 0.91 0.86 0.60 0 ]
      ; from Martin 1998, Tab. 4; first value (0) doesn't matter, as no
      ; mother mite invaded these cells
    [ set dataList [ 0 1 0.84 0.65 0.66 0 ] ]
 ]  ; drones (list length: 6)  [ 0 1 0.84 0.65 0.66 0 ] from Martin 1998, Tab. 4

 report item mitesIndex dataList

end

;
**********************************************************************************
**********************************************************************************
************

to-report  MiteOffspringREP [ ploidyMiteOrg ]
 ; reports offspring per mite depending on ploidy of bee brood and chosen reproduction model

 let result 0
 if ploidyMiteOrg != 1 and ploidyMiteOrg != 2
 [
  set BugAlarm true
  type "BUG ALARM in MiteOffspringREP! Wrong ploidyMiteOrg: "
  print ploidyMiteOrg
 ]

 if MiteReproductionModel = "Martin" or MiteReproductionModel = "Martin+0"
 [
  ifelse ploidyMiteOrg = 2
```

```
    [ set result 1.01 ]
      ; workers (1.01: Martin 1998; fertilisation already taken into account)

    [ set result 2.91 ]
 ]    ; drones (2.91: Martin 1998; fertilisation already taken into account)

 if MiteReproductionModel = "Fuchs&Langenbach"
 [
  ifelse ploidyMiteOrg = 2
    [ set result 1.4 * 0.95 ]
     ; workers (1.4: Fuchs&Langenbach 1989; of which 5% are
     ; unfertilised (Martin 1998 p.271))
    [ set result 2.21 * 0.967 ]
 ]  ; drones (2.21: Fuchs&Langenbach 1989; of which 3.3% are unfertilised (Martin 1998 p.271))

 if MiteReproductionModel = "No Mite Reproduction"  ; only for model testing
 [
  ifelse ploidyMiteOrg = 2
    [ set result 0 ]   ; workers
    [ set result 0 ]
 ] ; drones

 report result
end

;
********************************************************************************
********************************************************************************
************

; MitesReleaseProc: determines how many healthy and infected mites emerge from cells with a)
dead or b) emerging bees
; CALLED BY: WorkerLarvaeDevProc (dying), DroneLarvaeDevProc (dying), WorkerPupaeDevProc (2x,
for dying & emerging brood)
; DronePupaeDevProc (2x, for dying & emerging brood), BroodCareProc (4x, dying of drone & worker
larvae & pupae)

; .. all these procedures are called BEFORE the mite module (MiteProc)!

to MitesReleaseProc [ miteOrganiserID ploidyMiteOrg diedBrood releaseCausedBy ]
   ; 1. rate of healthy mites in the cellList 2. the relevant worker/drone
   ; cellListCondensed 3.  # died broodCells (0..n) 4. "emergingBrood" or "dyingBrood"

  let cellListCondensed []
   ; to not double the code for worker and drones, the local variable
   ; cellListCondensed is defined which stores EITHER the workerCellListCondensed
   ; OR the droneCellListCondensed

  let mitesInfectedSumUncappedCells 0
   ; sums up the infected mites of the current cohort
```

```
 let mitesHealthySumUncappedCells 0 ; sums up the healthy mites of the current cohort
 let mitesHealthy&InfectedSumUncappedCells 0
  ; sums up the healthy and infected mites of the current cohort

 let nPhoreticMitesBeforeEmergenceHealthy round (PhoreticMitesHealthyRate * PhoreticMites)
  ; saves the number healthy phoretic mites before the new mites emerge from their
  ; cells - necessary to calculate new PhoreticMitesHealthyRate

 let nPhoreticMitesBeforeEmergenceInfected PhoreticMites -
nPhoreticMitesBeforeEmergenceHealthy
  ; saves the number infected phoretic mites before the new mites emerge from
  ; their cells - necessary to calculate new PhoreticMitesHealthyRate

 let healthyRateMiteOrg 0
  ; proportion of healthy mites in the current cohort (miteOrganiser)

 let totalCells 0
  ; number of brood cells in the current cohort

 let releasedPupaeCohortsID -1

 let repetitions MAX_INVADED_MITES_WORKERCELL + 1
  ; to count the brood cells; (for worker cells); +1 as cells can also bee mite free
 if ploidyMiteOrg = 1
 [
  set repetitions MAX_INVADED_MITES_DRONECELL + 1
 ] ; ..the same for drone cells, +1 as cells can also bee mite free

 ; to save the required "cellListCondensed" and to determine the "who"
 ; of the affected (worker or drone) pupaeCohort:
 ask miteOrganisers with [ who = miteOrganiserID ]
 [
  ifelse ploidyMiteOrg = 1
   [
    set cellListCondensed droneCellListCondensed
     ; IF DRONES: local cellListCondensed =  droneCellListCondensed
    set releasedPupaeCohortsID invadedDroneCohortID
   ]  ; ... and affected droneCohort is the miteOrganisers "invadedDroneCohortID"
   [
    set cellListCondensed workerCellListCondensed
     ; ELSE WORKERS: local cellListCondensed = workerCellListCondensed
    set releasedPupaeCohortsID invadedWorkerCohortID
   ] ; ... and affected workerCohort is the miteOrganisers "invadedWorkerCohortID"
  set healthyRateMiteOrg invadedMitesHealthyRate
   ; saves the rate of healthy mites invaded to the current miteOrganiser
 ]

 let i 0
 repeat repetitions
 ; repetitions = MAX_INVADED_MITES_WORKER/DRONE_CELL + 1
 [
```

```
  ; counts the # of cells in the cellList
  set totalCells totalCells + (item i cellListCondensed)
  set i i + 1
]

let uncappedCells 0  ; number of cells that are uncapped ...
if releaseCausedBy = "dyingBrood" [ set uncappedCells diedBrood  ]
  ; .. because some pupae died..

if releaseCausedBy = "emergingBrood" [ set uncappedCells totalCells  ]
  ; .. or because all pupae emerge

if releaseCausedBy != "dyingBrood" and  releaseCausedBy != "emergingBrood"
[
  set BugAlarm true
  type "BUG ALARM in ReleaseMitesProc(1)! releaseCausedBy: "
  print releaseCausedBy
]  ; assertion

repeat uncappedCells
[
  ; uncapped brood cells are randomly chosen from all brood cells of
  ; this cohort. These cells may contain 0,1,2..invadedMitesCounter mites.
  ; These mother mites are released from the cell WITH OR WITHOUT
  ; reproduction and become phoretic

  let randomCell (random totalCells) + 1
    ; choses a random cell -> 1..totalCells (+1 as: random n = 0, 1, ..n-1)
    ; (totalCells is decreased at the end of each repetition by 1)

  let cellCounter 0
  let allMitesInSingleCell -1
    ; starting value of allMitesInSingleCell: -1 as it is increased by 1 in the
    ; following "while" loop
    ; allMitesInSingleCell: # of mites that invaded the randomly chosen cell

  while [ cellCounter < randomCell ]
    ; determines, by how many mites the "random cell"
    ; is invaded: sums up the # of cells invaded by 0 mites (1st loop)
    ; by 1 mite (2nd loop) etc. until the cellCounter >= randomCell
    ; the number of mites in random cell is then allMitesInSingleCell
  [
    set allMitesInSingleCell allMitesInSingleCell + 1
      ; in 1st loop: allMitesInSingleCell = 0! (i.e. item 0 = first item in list = 0 mites)
      ; in 2nd loop: 1 mite etc.

    set cellCounter cellCounter + (item allMitesInSingleCell cellListCondensed)
      ; cellCounter is increased by the # of cells with x mites in it
      ; (x = allMitesInSingleCell, i.e. 0,1,2..n)
  ]
```

```
; how many of the released mites are infected? -> 1. how many infected
; mites entered? 2. did they infect the larva? 3. how many healthy mites become
; infected by the infected larva?
let mitesIndex allMitesInSingleCell
  ; to address the correct item in the cellListCondensed after mite
  ; reproduction (i.e. when allMitesInSingleCell has changed)

let pupaInfected false ; a young larva is healthy
let infectedMitesInSingleCell 0
  ; the number of mites that were diseased on day of cell invasion

repeat allMitesInSingleCell
[
  ; invaded mites might be infected: repeat over all mites in the current brood cell
  if random-float 1 > healthyRateMiteOrg
  [
    set infectedMitesInSingleCell infectedMitesInSingleCell + 1
    ; this invaded mite was infected when invading the cell and is now counted as infected
  ]
]

let healthyMitesInSingleCell allMitesInSingleCell - infectedMitesInSingleCell
  ; healthy invaded mites are all invaded mites minus infected ones

if random-float 1 > (1 - VIRUS_TRANSMISSION_RATE_MITE_TO_PUPA) ^ infectedMitesInSingleCell
[
  set pupaInfected true
] ; as soon as at least 1 infected mite successfully infects the bee pupa, the bee pupa is infected

; PUPA ALIVE OR DEAD? (either died normally, died due to lack of nursing or killed by virus
let pupaAlive 1 ; (0 or 1) 1: = "yes", pupa is alive 0: = "no", pupa is dead
if pupaInfected = true
[
  if random-float 1 < VIRUS_KILLS_PUPA_PROB
  [
    set pupaAlive 0
  ]
] ; infected pupa might be killed by the virus. In this case:
  ; no offspring mites but still transmission of viruses to healthy mites in this cell
  ; (at least for DWV)

if releaseCausedBy = "dyingBrood"
[
  set pupaAlive 0
] ; larva/pupa is dead, if MitesReleaseProcis called, BECAUSE the brood died..

if releaseCausedBy = "emergingBrood" and allMitesInSingleCell > 0
[
  ; callow bees are emerging and with them the invaded mother mites and their offspring
  if pupaAlive = 0
  [
```

```
      ask turtles with [ who = releasedPupaeCohortsID ]
      [
        set number number - 1
          ; pupa died, hence the number of bees in this pupae cohort is reduced by 1
        set number_healthy number_healthy - 1
          ; pupa dies due to virus infection and has previously been healthy
        set Pupae_W&D_KilledByVirusToDay Pupae_W&D_KilledByVirusToDay + 1
      ]
    ]

    ; surviving but infected pupae:
    if pupaAlive = 1 and pupaInfected = true
    [
      ask turtles with [ who = releasedPupaeCohortsID ]
      [
        set number_infectedAsPupa number_infectedAsPupa + 1
        ; the bee was infected as pupa
        set number_healthy number_healthy - 1
        ; the pupa has become infected and is no longer healthy
      ]
    ]

    let averageOffspring
      random-poisson (MiteOffspringREP ploidyMiteOrg * MiteDensityFactorREP ploidyMiteOrg
mitesIndex)
      ; average # offspring of a single mother mite in the single cell (depends on ploidy of bee pupa
and # invaded mites)

    set healthyMitesInSingleCell allMitesInSingleCell
      * averageOffspring
        ; Offspring: all mites in cell x reprod. rate. NOTE: also infected mites
        ; may have healthy offspring! (MiteOffspringREP: reports # offspring for
        ; 1 mite in single invaded cell, for drones or workers)
      * pupaAlive
        ; pupaAlive =  1 or 0; if pupa is alive: normal mite reproduction, if dead:
        ; offspring = 0
      + healthyMitesInSingleCell          ; + mother mites

    set healthyMitesInSingleCell round healthyMitesInSingleCell
    ; this line is NOT NECESSARY as averageOffspring is integer!
    set allMitesInSingleCell healthyMitesInSingleCell + infectedMitesInSingleCell
      ; update of total mites in the cell
  ]  ; END of "if releaseCausedBy = 'emergingBrood' "

  if pupaAlive = 1 and pupaInfected = true
  [
    ; if the bee pupa was infected by an infected mite AND IS STILL ALIVE,
    ; then the healthy mites (invaded or offspring) might become infected too

    repeat healthyMitesInSingleCell
    [
```

```
    ; all healthy mites have then the risk to become infected too
    if random-float 1 < VIRUS_TRANSMISSION_RATE_PUPA_TO_MITES
    ; if random number < the transmission rate from bee pupa to mite, the healthy
    ; mite becomes infected
    [
      set healthyMitesInSingleCell healthyMitesInSingleCell - 1
        ; hence: the number of healthy released mites decreases by 1..

      set infectedMitesInSingleCell infectedMitesInSingleCell + 1
    ]  ; .. and the number of infected released mites increases by 1
  ] ; end of 'repeat sumInvadedMitesHealthy'
] ; end of 'IF pupaInfected' - now the numbers of healthy and infected (mother) mites in
; single cell is known (= healthyMitesInSingleCell and infectedMitesInSingleCell)

if healthyMitesInSingleCell + infectedMitesInSingleCell != allMitesInSingleCell
[
  set BugAlarm true
  type "BUG ALARM in ReleaseMitesProc(2)! allMitesInSingleCell: "
  type allMitesInSingleCell
  type " infectedMitesInSingleCell: "
  type infectedMitesInSingleCell
  type " healthyMitesInSingleCell: "
  print healthyMitesInSingleCell
]

; MITE FALL:
let miteFallProb MITE_FALL_DRONECELL
if ploidyMiteOrg = 2
[
  set miteFallProb MITE_FALL_WORKERCELL
] ; probabilities of mites to fall from comb, depending on cell type

repeat healthyMitesInSingleCell
[ ; determined for healthy and infected mites separately
  if random-float 1 < miteFallProb
  [
    set healthyMitesInSingleCell healthyMitesInSingleCell - 1
    set allMitesInSingleCell allMitesInSingleCell - 1
    set DailyMiteFall DailyMiteFall + 1
  ]
]

repeat infectedMitesInSingleCell
[
  if random-float 1 < miteFallProb
  [
    set infectedMitesInSingleCell infectedMitesInSingleCell - 1
    set allMitesInSingleCell allMitesInSingleCell - 1
    set DailyMiteFall DailyMiteFall + 1
  ]
]
```

```
    set mitesHealthySumUncappedCells mitesHealthySumUncappedCells + healthyMitesInSingleCell
      ; sums up all healthy mites emerging from current cohort
      ; (set to 0 at beginning of this procedure)

    set mitesInfectedSumUncappedCells mitesInfectedSumUncappedCells + infectedMitesInSingleCell
      ; same for infected mites (set to 0 at beginning of this procedure)

    set PhoreticMites PhoreticMites + allMitesInSingleCell
      ; mother mites in this uncapped brood cell are released from the brood
      ; cell and become phoretic..

    set mitesHealthy&InfectedSumUncappedCells
      mitesHealthy&InfectedSumUncappedCells + allMitesInSingleCell
        ; released mites from all brood cell in this cohort are totaled up

    set cellListCondensed replace-item mitesIndex cellListCondensed
      (item mitesIndex cellListCondensed - 1)
        ; .. and one brood cell is removed; mitesIdex: number of mother mites that
        ; invaded the brood cell

    if item mitesIndex cellListCondensed < 0
    [
      set BugAlarm true
      type "BUG ALARM in ReleaseMitesProc(3)! Negative number in cellListCondensed
(releaseMitesProc)! "
      show cellListCondensed
    ]

    set totalCells totalCells - 1
      ; number of total brood cells in this cohort is reduced by 1

    if totalCells < 0
    [
      set BugAlarm true
      type "BUG ALARM in ReleaseMitesProc(4)! Negative number of  total cells in releaseMitesProc: "
      print totalCells
    ]
  ] ; END OF "REPEAT UNCAPPEDCELLS"

  set NewReleasedMitesToday
    NewReleasedMitesToday + mitesHealthy&InfectedSumUncappedCells
      ; # of newly released (mother+offspring) mites (only those that survived
      ; MiteFall) is summed up (set to 0 in DailyUpdateProc)

  if mitesInfectedSumUncappedCells + mitesHealthySumUncappedCells
     != mitesHealthy&InfectedSumUncappedCells
  [ ; assertion
    set BugAlarm true
    type "BUG ALARM in ReleaseMitesProc(5)! mitesInfectedSumUncappedCells: "
    type mitesInfectedSumUncappedCells
```

```
  type " mitesHealthySumUncappedCells: "
  type mitesHealthySumUncappedCells
  type " mitesHealthy&InfectedSumUncappedCells: "
  print mitesHealthy&InfectedSumUncappedCells
 ]

 if mitesInfectedSumUncappedCells < 0 or mitesHealthySumUncappedCells < 0
 [ ; assertion
  set BugAlarm true
  type "BUG ALARM in ReleaseMitesProc(6)! mitesInfectedSumUncappedCells: "
  type mitesInfectedSumUncappedCells
  type " mitesHealthySumUncappedCells: "
  type mitesHealthySumUncappedCells
  type " mitesHealthy&InfectedSumUncappedCells: "
  print mitesHealthy&InfectedSumUncappedCells
 ]

; Updating of the actual cell lists - either for the drone or for the worker brood:
 ask miteOrganisers with [ who = miteOrganiserID ]
 [ ; assertion
  if ploidyMiteOrg = 1 [ set droneCellListCondensed cellListCondensed ]    ; IF drones
  if ploidyMiteOrg = 2 [ set workerCellListCondensed cellListCondensed ]    ; IF workers
  if (ploidyMiteOrg != 1) and (ploidyMiteOrg != 2)
  [
   set BugAlarm true
   type "BUG ALARM in releaseMitesProc(7)! Wrong ploidyMiteOrg: "
   print ploidyMiteOrg
  ]
                                    ]
 ; UPDATE of the healthy mite rate:
 if ( nPhoreticMitesBeforeEmergenceHealthy
     + nPhoreticMitesBeforeEmergenceInfected
     + mitesHealthySumUncappedCells
     + mitesInfectedSumUncappedCells) > 0
 [
  set PhoreticMitesHealthyRate
    ( nPhoreticMitesBeforeEmergenceHealthy + mitesHealthySumUncappedCells)
      / ( nPhoreticMitesBeforeEmergenceHealthy
         + nPhoreticMitesBeforeEmergenceInfected
         + mitesHealthySumUncappedCells
         + mitesInfectedSumUncappedCells )
 ]

 end

;
**********************************************************************************
**********************************************************************************
************

to MiteDailyMortalityProc
```

```
 ifelse ( TotalEggs + TotalLarvae
        + TotalPupae + TotalDroneEggs
        + TotalDroneLarvae + TotalDronePupae) > 0 ; is it within brood period?
   [
    set PhoreticMites
     (PhoreticMites - random-poisson (PhoreticMites *  MITE_MORTALITY_BROODPERIOD))
   ]  ; IF brood is present
   [
    set PhoreticMites
    (PhoreticMites - random-poisson (PhoreticMites *  MITE_MORTALITY_WINTER))
   ] ; ELSE: if no brood is present
end

;
**********************************************************************************
**********************************************************************************
************

to MitePhoreticPhaseProc
 ; infection of healthy worker bees via infected phoretic mites and of
 ; healthy phoretic mites via infected workers; Called daily by MiteProc

 let healthyPhoreticMites round (PhoreticMites * PhoreticMitesHealthyRate)
  ; # of healthy, phoretic mites is calculated from the rate of healthy phoretic mites

 let infectedPhoreticMites PhoreticMites - healthyPhoreticMites
  ; all other phoretic mites are infected

 let phoreticMitesPerIHbee 0

 if ( TotalIHbees + InhivebeesDiedToday
    + NewForagerSquadronsHealthy
    + NewForagerSquadronsInfectedAsPupae
    + NewForagerSquadronsInfectedAsAdults > 0 ) ; avoid division by 0
 [
  set phoreticMitesPerIHbee
    ( PhoreticMites - NewReleasedMitesToday)
     / (TotalIHbees + InhivebeesDiedToday
       + SQUADRON_SIZE *
         ( NewForagerSquadronsHealthy
           + NewForagerSquadronsInfectedAsPupae
           + NewForagerSquadronsInfectedAsAdults
         )
       )
 ] ; phoretic mites are assumed to infest only inhive bees,
  ; "ih-bees" here = current ih-bees + ih-bees died today
  ;              + ih-bees developed into foragers today!

; mites are released from inhive bees, if ih-bees die or develop into foragers:
 let mitesReleasedFromInhivebees
  precision
```

```
      (
       phoreticMitesPerIHbee
        * ( InhivebeesDiedToday  ; died ih-bees
           + SQUADRON_SIZE      ; new foragers:
            * ( NewForagerSquadronsHealthy
               + NewForagerSquadronsInfectedAsPupae
               + NewForagerSquadronsInfectedAsAdults
             )
          )
      ) 5

  if mitesReleasedFromInhivebees > PhoreticMites
  [
    set BugAlarm true
    type "BugAlarm!!! mitesReleasedFromInhivebees > PhoreticMites! mitesReleasedFromInhivebees:
"
    type mitesReleasedFromInhivebees
    type " PhoreticMites: "
    print PhoreticMites
  ]

  let healthyPhoreticMitesSwitchingHosts
    round
      (
        mitesReleasedFromInhivebees * PhoreticMitesHealthyRate
       + PhoreticMites * PropNewToAllPhorMites * PhoreticMitesHealthyRate
      )  ; # healthy phoretic mites that infest a bee. These are: newly
         ; released mites that haven't entered a brood cell (hence:
         ; "phoreticMites * PropNewToAllPhorMites") and phoretic mites, where the host
         ; bee just died; all multiplied with PhoreticMitesHealthyRate as only healthy
         ; mites are considered

  if healthyPhoreticMitesSwitchingHosts > healthyPhoreticMites
  [
    ; set BugAlarm true
    if (healthyPhoreticMitesSwitchingHosts - healthyPhoreticMites) > 1
    [
      set BugAlarm true  ; if difference > 1 it can't be explained by rounding errors..
      type "BugAlarm!!! (MitePhoreticPhaseProc)  healthyPhoreticMitesSwitchingHosts >
healthyPhoreticMites! healthyPhoreticMitesSwitchingHosts: "
      type healthyPhoreticMitesSwitchingHosts
      type " healthyPhoreticMites: "
      print healthyPhoreticMites
    ]

    set healthyPhoreticMitesSwitchingHosts healthyPhoreticMites
  ] ; to ensure that not more mites switch their hosts than actually present!

  ; healthy and infected IN-HIVE bees:
  let totalInfectedWorkers 0
  let totalHealthyWorkers 0
```

```
 ask IHbeeCohorts
 [
  set totalInfectedWorkers
   totalInfectedWorkers + number_infectedAsPupa + number_infectedAsAdult
    ; infected: either during pupal phase or as adults
  set totalHealthyWorkers totalHealthyWorkers + number_healthy
 ]

 ; Infection of healthy mites:
 let newlyInfectedMites 0
  ; the probability of healthy mites to become infected equals the proportion of
  ; infected in-hive workers to all in-hive workers:
 if (totalInfectedWorkers + totalHealthyWorkers) > 0  ; avoid division by 0!
  [
  repeat healthyPhoreticMitesSwitchingHosts
   [
    if random-float 1 <  totalInfectedWorkers / (totalInfectedWorkers + totalHealthyWorkers)
     [
      set newlyInfectedMites newlyInfectedMites + 1
     ]
   ]
  ]

 ; infection of healthy adult workers - ONLY IN-HIVE WORKERS!
 let allInfectedMitesSwitchingHosts
  round
   ( PhoreticMites * PropNewToAllPhorMites * (1 - PhoreticMitesHealthyRate)
    + mitesReleasedFromInhivebees * (1 - PhoreticMitesHealthyRate))
    ; # infected phoretic mites that infest a new bee. These are: newly
    ; released mites, that haven't entered a brood cell (hence: "phoreticMites
    ; * PropNewToAllPhorMites") and phoretic mites, where the host bee just died;
    ; all multiplied with (1 - PhoreticMitesHealthyRate) as only infected mites are considered

 ask IHbeeCohorts
 [
  if TotalIHbees > 0 and number > 0  ; avoid division by 0!
  [
   let infectedMitesSwitchingHostsInThisCohort
    (allInfectedMitesSwitchingHosts / TotalIHbees) *  number
     ; # of infected mites switching their host in current bee cohort: # mites per ih-bee * number of
ih-bees
     ; in this cohort (assumes an equal distribution of mites)

   let newlyInfectedIHbeesInThisCohort 0
   repeat number_healthy  ; only healthy bees can become newly infected
    [
     if random-float 1 > (1 - (1 / number)) ^ infectedMitesSwitchingHostsInThisCohort
     ; "number" (i.e. all bees in this cohort) as mites can also jump on already infected bees
      [
       set newlyInfectedIHbeesInThisCohort newlyInfectedIHbeesInThisCohort + 1
        ; # of newly infected bees is increased by 1
```

```
          set infectedMitesSwitchingHostsInThisCohort infectedMitesSwitchingHostsInThisCohort - 1
          if infectedMitesSwitchingHostsInThisCohort < 0
            [ set infectedMitesSwitchingHostsInThisCohort 0 ]
        ]
      ]


    ; Assertion to be sure there are not more newly infected bees than there were healthy bees:
    if newlyInfectedIHbeesInThisCohort > number_healthy
    [
     set BugAlarm true
     print "Bug Alarm! newlyInfectedIHbeesInThisCohort > number_healthy!"

    ]

    set number_infectedAsAdult number_infectedAsAdult + newlyInfectedIHbeesInThisCohort
    set number_healthy  number_healthy - newlyInfectedIHbeesInThisCohort

    if number_healthy < 0
    [
     set BugAlarm true
     type "BUG ALARM!!! (MitePhoreticPhaseProc) Negative number of healthy IH bees
(MitePhoreticPhaseProc): "
     show number_healthy
    ]

    if number_healthy + number_infectedAsPupa + number_infectedAsAdult != number
    [
     set BugAlarm true
     type "BUG ALARM!!! (MitePhoreticPhaseProc) Wrong sum of healthy + infected bees in this
cohort: "
     type number_healthy + number_infectedAsPupa + number_infectedAsAdult
     type " instead of: "
     show number
    ]
   ]  ; end "if TotalIHbees > 0 and number > 0 "
  ] ; end "ask IHbeeCohorts "

  set infectedPhoreticMites infectedPhoreticMites + newlyInfectedMites
  set healthyPhoreticMites healthyPhoreticMites - newlyInfectedMites

  if healthyPhoreticMites < 0
  [
   set BugAlarm true
   type "BUG ALARM!!! Negative number of healthy mites (MitePhoreticPhaseProc): "
   show healthyPhoreticMites
  ]

  if infectedPhoreticMites + healthyPhoreticMites > 0
  [
   set PhoreticMitesHealthyRate
```

```
    healthyPhoreticMites / (infectedPhoreticMites + healthyPhoreticMites)
  ]

end

;
**********************************************************************************
**********************************************************************************
************

to MiteOrganisersUpdateProc
  set TotalMites 0
    ; all mites in the colony, irrespective if phoretic or in cells

  ask miteOrganisers
  [
    back 1 ; new position in the GUI
    set age age + 1
    set cohortInvadedMitesSum 0
    let counter 0
      ; counts total numbers of mites in brood cells for each miteOrganiser (="mite cohort")

    foreach workerCellListCondensed
    [
      set cohortInvadedMitesSum cohortInvadedMitesSum + (? * counter)
      set counter counter + 1
    ] ; sums up the mites in worker cells ( multiplication of # cells with X mites in them * X) (X =
counter)

    set counter 0

    foreach droneCellListCondensed
    [
      set cohortInvadedMitesSum cohortInvadedMitesSum
         + (? * counter)
      set counter counter + 1
    ] ; sums up the mites in drone cells ( multiplication of # cells with X mites in them * X) (X =
counter)

    set label cohortInvadedMitesSum
    set TotalMites TotalMites + cohortInvadedMitesSum
      ; interim result: summing up all the mites in the cells

    if (age > DRONE_EMERGING_AGE) and (age >= EMERGING_AGE)
    [
      die
    ]  ; ">" (not ">=") as they age at the beginning of this procedure
  ] ; end "ask miteOrganisers "

 set TotalMites TotalMites + PhoreticMites
   ; final result: TotalMites = all mites in the cells + phoretic mites
```

```
end

;
**********************************************************************************
**********************************************************************************
************

; ..............  END OF THE VARROA MITE SUBMODEL  ..........................................  END
OF THE VARROA MITE SUBMODEL  ..................


;
**********************************************************************************
**********************************************************************************
************

to CountingProc
  ; counts # bees in different stages, castes CALLED BY: 1. BroodCareProc 2. Go 3. MiteProcedure

  ; WORKERS:
  set TotalEggs 0 ask eggCohorts [ set TotalEggs (TotalEggs + number)]
  set TotalLarvae 0 ask larvaeCohorts [ set TotalLarvae (TotalLarvae + number)]
  set TotalPupae 0 ask pupaeCohorts [ set TotalPupae (TotalPupae + number)]
  set TotalIHbees 0 ask IHbeeCohorts [ set TotalIHbees (TotalIHbees + number)]
  set TotalForagers (count foragerSquadrons) * SQUADRON_SIZE

  ; DRONES:
  set TotalDroneEggs 0 ask DroneEggCohorts [ set TotalDroneEggs (TotalDroneEggs + number)]
  set TotalDroneLarvae 0 ask DroneLarvaeCohorts [ set TotalDroneLarvae (TotalDroneLarvae +
number)]
  set TotalDronePupae 0 ask DronePupaeCohorts [ set TotalDronePupae (TotalDronePupae +
number)]
  set TotalDrones 0 ask DroneCohorts [ set TotalDrones (TotalDrones + number)]
  set TotalWorkerAndDroneBrood TotalEggs + TotalLarvae + TotalPupae + TotalDroneEggs +
TotalDroneLarvae + TotalDronePupae
  if TotalEggs  < 0 OR TotalLarvae < 0 OR TotalPupae < 0 OR TotalIHbees < 0 OR TotalForagers < 0
  [
    set BugAlarm true
    output-show (word ticks " BUG ALARM! negative number in total bees")
    type "TotalEggs: "
    type TotalEggs
    type " TotalLarvae: "
    type TotalLarvae
    type " TotalPupae: "
    type TotalPupae
    type " TotalIHbees: "
    type TotalIHbees
    type " TotalForagers: "
    print TotalForagers
  ]
```

```
  ask turtles
  [
    if number < 0
    [
      set BugAlarm true
      type (word ticks " BUG ALARM! negative number in turtles: ")
      show number
      ]
  ]

  if TotalMites < 0 or PhoreticMites < 0 or PhoreticMitesHealthyRate > 1  or
PhoreticMitesHealthyRate < 0
  [
    set BugAlarm true
    output-show (word ticks " BUG ALARM! Check number of mites and PhoreticMitesHealthyRate!")
    type "PhoreticMitesHealthyRate: "
    type PhoreticMitesHealthyRate
    type " TotalMites: "
    type TotalMites
    type " PhoreticMites: "
    type PhoreticMites
  ]

  ask (turtle-set pupaeCohorts dronePupaeCohorts droneCohorts)
  [
    if number != number_infectedAsPupa + number_healthy
    [
      set BugAlarm true
      show "BUG ALARM! (CountingProc) number <> healthy + infected"
    ]
  ]

  ask IHbeeCohorts
  [
    if number != number_infectedAsAdult + number_infectedAsPupa + number_healthy
    [
      set BugAlarm true
      show "BUG ALARM! (CountingProc) number <> healthy + infected (IH-bees)"
    ]
  ]
end

;
*******************************************************************************
*******************************************************************************
************

to PollenConsumptionProc
  ; calculates the daily pollen consumption

  let DAILY_POLLEN_NEED_ADULT 1.5 ; 0 ;1.5 ; 1.5 ;
```

```
   ; 1.5 mg fresh pollen per Day per bee (based on
   ; Pernal, Currie 2000, value for 14d old bees, Fig. 3)

  let DAILY_POLLEN_NEED_ADULT_DRONE 2 ; just an ESTIMATION

  let DAILY_POLLEN_NEED_LARVA 142 / (PUPATION_AGE - HATCHING_AGE)
   ; (23.6 mg/d) see HoPoMo
  let DAILY_POLLEN_NEED_DRONE_LARVA 50
   ; ESTIMATION, Rortais et al. 2005: "The pollen consumption of drone larvae has never been
determined."

  let pollenStoreLasting_d 7
   ; similar to "FACTORpollenstorage" of HoPoMo model, which is set to 6.
   ; Seeley 1995: pollen stores last for about 1 week;

  let needPollenAdult
   ((TotalIHbees + TotalForagers) *  DAILY_POLLEN_NEED_ADULT
    + TotalDrones * DAILY_POLLEN_NEED_ADULT_DRONE )

  let needPollenLarvae (TotalLarvae * DAILY_POLLEN_NEED_LARVA
    + TotalDroneLarvae * DAILY_POLLEN_NEED_DRONE_LARVA )

  set DailyPollenConsumption_g (needPollenAdult + needPollenLarvae) / 1000 ; [g]
  set PollenStore_g PollenStore_g - DailyPollenConsumption_g
  if PollenStore_g < 0
  [
   set PollenStore_g 0
  ]

  ; the amount of pollen a colony tries to keep (depends on its current pollen consumption):
  set IdealPollenStore_g DailyPollenConsumption_g * pollenStoreLasting_d ; [g]

  if IdealPollenStore_g < MIN_IDEAL_POLLEN_STORE
  [
   set IdealPollenStore_g MIN_IDEAL_POLLEN_STORE
  ]

  ; PollenIdeal: switch in GUI, if true: pollen stores are always "ideal":
  if PollenIdeal = true
  [
   set PollenStore_g IdealPollenStore_g
  ]

  ; if no more pollen is left, protein stores of nurse bees are reduced.
  ;Assumption: protein stores of nurses can last for 7d, if the max. amount of brood (rel. to # nurses)
is present, or proportionally longer if less brood is present:
  let workloadNurses 0
  if (TotalIHbees + TotalForagers * FORAGER_NURSING_CONTRIBUTION) *
MAX_BROOD_NURSE_RATIO > 0
  [
   set workloadNurses
```

```
    TotalWorkerAndDroneBrood /
     ((TotalIHbees + TotalForagers * FORAGER_NURSING_CONTRIBUTION) *
MAX_BROOD_NURSE_RATIO)
 ]

 ifelse PollenStore_g > 0
  [
   set ProteinFactorNurses ProteinFactorNurses + (1 / PROTEIN_STORE_NURSES_d)
  ] ;  IF pollen in present in colony, nurses can restore the protein stores of
   ; their bodies (within 7d)
  [
   set ProteinFactorNurses ProteinFactorNurses - (workloadNurses / PROTEIN_STORE_NURSES_d)
  ] ; ELSE protein content of brood food decreases, depending on brood to nurse ratio

 if ProteinFactorNurses > 1 [ set ProteinFactorNurses 1 ]
  ; range of ProteinFactorNurses between 1..

 if ProteinFactorNurses < 0 [ set ProteinFactorNurses 0 ]  ; .. and 0
end

;
**********************************************************************************
**********************************************************************************
************

to HoneyConsumptionProc
 let DAILY_HONEY_NEED_ADULT_RESTING 11 ; 15 ; (11)
  ; [mg/Day of honey] Rortais et al 2005: Winter bees: 11 mg/d (based on
  ; assumptions from Winston, 1987)

 let DAILY_HONEY_NEED_NURSES 53.42  ; (53.42) [mg/Day of honey]
  ; Rortais et al 2005: average for "brood attending" 34-50mg sugar/d => 43-64mg/d honey

 let THERMOREGULATION_BROOD (DAILY_HONEY_NEED_NURSES -
DAILY_HONEY_NEED_ADULT_RESTING)
  / MAX_BROOD_NURSE_RATIO
  ;  additional cost per broodcell (e.g. Thermoregulation): difference between nursing
  ; and resting divided by # broodcells;

 let DAILY_HONEY_NEED_LARVA 65.4 / (PUPATION_AGE - HATCHING_AGE) ; [mg/day]
  ; = 10.9[mg] HONEY per Day per larvae  = 163.5mg nectar in total * 0.4
  ; (0.4: Nectar to honey); HoPoMo =  65.4 mg / 6

 let DAILY_HONEY_NEED_DRONE_LARVA 19.2 ;
  ; [mg/Day of honey] Rortais et al 2005: 98.2mg sugar in 6.5d
  ; sugar to honey: x1.272 i.e. 124.9mg honey in total or 19.2 mg/d

 let DAILY_HONEY_NEED_ADULT_DRONE 10  ;
  ; (9.806 = 10mg honey per day): Winston p62: resting drone 1-3mg sugar/hr
  ; flying drone: 14mg/hr (Mindt 1962); assumptions: 22h resting, 2h flying (MB);
  ; 1 mg sucrose = 17J; 1kJ = 0.008013g Honig
```

```
  ; honey costs of all adults, in-hive bees, foragers and drones:
  let needHoneyAdult
     (TotalIHbees + TotalForagers) * DAILY_HONEY_NEED_ADULT_RESTING
     + TotalDrones * DAILY_HONEY_NEED_ADULT_DRONE

  let needHoneyLarvae
     TotalLarvae * DAILY_HONEY_NEED_LARVA
      + TotalDroneLarvae * DAILY_HONEY_NEED_DRONE_LARVA

  set DailyHoneyConsumption
    needHoneyAdult + needHoneyLarvae + TotalWorkerAndDroneBrood
     * THERMOREGULATION_BROOD  ; [mg]

  ; the honey consumption is removed from the honey stores:
  set HoneyEnergyStore
     HoneyEnergyStore
     - (DailyHoneyConsumption / 1000) * ENERGY_HONEY_per_g

  ; sum up the total honey consumption as potential output:
  set CumulativeHoneyConsumption
    CumulativeHoneyConsumption  + DailyHoneyConsumption  ;[mg]

  ; HoneyIdeal: switch in GUI, if true: honey stores are always full:
  if HoneyIdeal = true
  [
   set HoneyEnergyStore MAX_HONEY_ENERGY_STORE
  ]
end

;
*********************************************************************************
*********************************************************************************
************

to BeekeepingProc
  let winterPauseStart 320 ; 320 = mid November
  let winterPauseStop 45 ; 45 = mid February
  let minWinterStore_kg 16 ; [kg] honey
  let minSummerStore_kg 3   ; [kg]
  let addedFondant_kg 1 ; [kg]
  ;let addedPollen_kg 0.5  ; [kg]

  ; FEEDING OF COLONY:
  ask Signs with [shape = "ambrosia"] [ hide-turtle]

  if FeedBees = true
    and day < winterPauseStart
    and day > winterPauseStop
    and HoneyEnergyStore / ( ENERGY_HONEY_per_g * 1000 ) < minSummerStore_kg
      ; feeding colony in spring or summer
```

```
  [
    set TotalHoneyFed_kg TotalHoneyFed_kg + addedFondant_kg
    set HoneyEnergyStore HoneyEnergyStore + (addedFondant_kg * ENERGY_HONEY_per_g * 1000)
    output-type "Feeding colony on day "
    output-type ceiling (day mod 30.4374999) ; day
    output-type "."
    output-type floor(day / (365.25 / 12)) + 1 ; month
    output-type "."
    output-type ceiling (ticks / 365)     ; year
    output-type " Fondant provided [kg]: "
    output-type precision addedFondant_kg 1
    output-type " total food added [kg]: "
    output-print precision TotalHoneyFed_kg 1
    ask Signs with [shape = "ambrosia"] [ show-turtle]
  ]

  if FeedBees = true
    and day = winterPauseStart
    and HoneyEnergyStore / ( ENERGY_HONEY_per_g * 1000 ) < minWinterStore_kg
      ; feeding colony before winter
  [
    set TotalHoneyFed_kg TotalHoneyFed_kg
      + minWinterStore_kg
      -(HoneyEnergyStore / ( ENERGY_HONEY_per_g * 1000 ))

    output-type "Feeding colony on day "
    output-type day
    output-type ". Ambrosia fed [kg]: "
    output-type precision (minWinterStore_kg - (HoneyEnergyStore / ( ENERGY_HONEY_per_g * 1000
))) 1
    output-type " total food added [kg]: "
    output-print precision TotalHoneyFed_kg 1
    set HoneyEnergyStore minWinterStore_kg * 1000 * ENERGY_HONEY_per_g
      ; if honey store is smaller than minWinterStore it is filled up to minWinterStore

    ask Signs with [shape = "ambrosia"] [ show-turtle]
  ]

  ; ADD BEES TO WEAK COLONY - a weak colony is "merged" with another
  ; (not modelled!) weak colony (all of them are healthy):
  ask signs with [shape = "colonies_merged"] [ hide-turtle ]
  if MergeWeakColonies = true
    and (TotalIHbees + TotalForagers) < MergeColoniesTH
    and day = winterPauseStart
  [
    set TotalBeesAdded TotalBeesAdded + MergeColoniesTH
    output-type "Merging colonies in autumn! "
    output-type " # added bees: "
    output-type MergeColoniesTH
    output-type " total bees added: "
    output-print TotalBeesAdded
```

```
   ask signs with [shape = "colonies_merged"] [ show-turtle ]

   create-foragerSquadrons (MergeColoniesTH / SQUADRON_SIZE)
   [
    set age 60 + random 40
    setxy 30 9
    set color grey
    set size 2
    set heading 90
    set shape "bee_mb_1"
    set mileometer random (MAX_TOTAL_KM / 5)
    set activity "resting"
    set activityList [ ]
    set cropEnergyLoad 0 ; [kJ] no nectar in the crop yet
    set collectedPollen 0 ; [g] no pollen pellets
    set knownNectarPatch -1  ; -1 = no nectar Flower patch known
    set knownPollenPatch -1 ; -1 = no pollen Flower patch known
    set pollenForager false ; new foragers are nectar foragers
    set infectionState "healthy"
      ; possible infection states are: "healthy" "infectedAsPupa" "infectedAsAdult"
   ]
  ]  ; if MergeWeakColonies = true  ...

  ; ADDING POLLEN IN SPRING:
  ask signs with [shape = "pollengrain"] [ hide-turtle ]
  if AddPollen = true and day = 90 ; day 90: end of March
  [
   ask signs with [shape = "pollengrain"] [ show-turtle ]
   set TotalPollenAdded TotalPollenAdded + addedPollen_kg
   output-type "Added pollen [kg]: "
   output-type addedPollen_kg
   output-type " total pollen added [kg]: "
   output-print TotalPollenAdded
   set PollenStore_g PollenStore_g + addedPollen_kg * 1000
  ]

  ask Signs with [shape = "honeyjar"] [ hide-turtle ]
  if ((Day >=  HarvestingDay)
   and (Day <  HarvestingDay + HarvestingPeriod)
   and (HoneyHarvesting = true))
     ; honey can only be harvested within HarvestingPeriod
  [
   if HoneyEnergyStore / ( ENERGY_HONEY_per_g * 1000 ) > HarvestingTH
   [
    set HarvestedHoney_kg (HoneyEnergyStore  / (ENERGY_HONEY_per_g * 1000)) -
RemainingHoney_kg
    set HoneyEnergyStore HoneyEnergyStore - (HarvestedHoney_kg * ENERGY_HONEY_per_g *
1000)
    set TotalHoneyHarvested_kg TotalHoneyHarvested_kg + HarvestedHoney_kg
    output-type "Honey harvest on day "
    output-type ceiling (day mod 30.4374999)
```

```
    output-type "."
    output-type floor(day / (365.25 / 12)) + 1
    output-type "."
    output-type ceiling (ticks / 365)
    output-type ". Amount [kg]: "
    output-type precision HarvestedHoney_kg 1
    output-type " total honey harvested: "
    output-print precision TotalHoneyHarvested_kg 1
    ask Signs with [shape = "honeyjar"]
    [
     show-turtle
     set label precision HarvestedHoney_kg 1
    ]
   ]
  ]

  if QueenAgeing = true
  [
   let requeening true ; true
   if requeening = true and Queenage >= 375
   [
    set Queenage 10
    output-print word "New queen inserted on day " day
   ]  ; old queen is replaced by the beekeeper
  ]

- ; begin ***NEW FOR BEEHAVE_BEEMAPP2015***
  ; let treatmentDay 270   ; 270: 27.September
  ; let treatmentDuration 40  ; (28-40d) Fries et al. 1994
  ; let treatmentEfficiency 0.115
   ; (0.115) Fries et al. 1994 kills X*100% of phoretic mites each treatment Day

  ; treatment #1:
  if EfficiencyPhoretic > 1 [ set EfficiencyPhoretic 1 ]
  ifelse ((varroaTreatment = true) and (Day >= treatmentDay)
    and (Day <=  treatmentDay + treatmentDuration ~~)~~))
  ~~  and (N_INITIAL_MITES_HEALTHY + N_INITIAL_MITES_INFECTED > 0))~~
    [
     set PhoreticMites round(PhoreticMites * (1 - ~~treatmentEfficiency~~EfficiencyPhoretic))
     ask signs with [shape = "x" or shape = "varroamite03"] [ show-turtle]



    if KillOpenBrood = true
     [
      ask (turtle-set eggCohorts larvaeCohorts) with [ age < PUPATION_AGE ] [ set number 0 ]
      ask (turtle-set droneEggCohorts droneLarvaeCohorts) with [ age < DRONE_PUPATION_AGE ] [
set number 0 ]
      ask miteOrganisers with [ age <= 10 ] ; i.e. those mite organisers, connected to dying larvae
cohorts
      [
```

```
        if age < 10 ; for workers: age 10 brood is already capped, i.e. not affected!
          [ set workerCellListCondensed n-values (MAX_INVADED_MITES_WORKERCELL + 1) [ 0 ]]
        set droneCellListCondensed n-values (MAX_INVADED_MITES_DRONECELL + 1) [ 0 ]
        let memoInvadedW invadedWorkerCohortID
        let memoInvadedD invadedDroneCohortID
        if any? turtles with [ who = memoInvadedW ] [ set workerCellListCondensed replace-item 0
workerCellListCondensed [number] of turtle invadedWorkerCohortID ]
        if any? turtles with [ who = memoInvadedD ] [ set droneCellListCondensed replace-item 0
droneCellListCondensed [number] of turtle invadedDroneCohortID ]
      ]
    ]

    if KillAllMitesInCells = true
    [
      ask miteOrganisers
      [
        set workerCellListCondensed n-values (MAX_INVADED_MITES_WORKERCELL + 1) [ 0 ]
        set droneCellListCondensed n-values (MAX_INVADED_MITES_DRONECELL + 1) [ 0 ]
        let memoInvadedW invadedWorkerCohortID
        let memoInvadedD invadedDroneCohortID
        if any? turtles with [ who = memoInvadedW ] [ set workerCellListCondensed replace-item 0
workerCellListCondensed [number] of turtle invadedWorkerCohortID ]
        if any? turtles with [ who = memoInvadedD ] [ set droneCellListCondensed replace-item 0
droneCellListCondensed [number] of turtle invadedDroneCohortID ]
      ]
    ]
  ]
  [
    ask signs with [shape = "x" or shape = "varroamite03"] [ hide-turtle]
  ]

  ; treatment #2:
  if EfficiencyPhoretic2 > 1 [ set EfficiencyPhoretic2 1 ]
  if ((varroaTreatment = true) and (Day >= treatmentDay2)
    and (Day <=  treatmentDay2 + treatmentDuration2 ))
    [
      set PhoreticMites round (PhoreticMites * (1 - EfficiencyPhoretic2))
      ask signs with [shape = "x" or shape = "varroamite03"] [ show-turtle]
      if KillOpenBrood2 = true
      [
        ask (turtle-set eggCohorts larvaeCohorts) with [ age < PUPATION_AGE ] [ set number 0 ]
        ask (turtle-set droneEggCohorts droneLarvaeCohorts) with [ age < DRONE_PUPATION_AGE ] [
set number 0 ]
        ask miteOrganisers with [ age <= 10 ] ; i.e. those mite organisers, connected to dying larvae
cohorts
        [
          if age < 10 ; for workers: age 10 brood is already capped, i.e. not affected!
            [ set workerCellListCondensed n-values (MAX_INVADED_MITES_WORKERCELL + 1) [ 0 ]]
          set droneCellListCondensed n-values (MAX_INVADED_MITES_DRONECELL + 1) [ 0 ]
          let memoInvadedW invadedWorkerCohortID
          let memoInvadedD invadedDroneCohortID
```

```
        if any? turtles with [ who = memoInvadedW ] [ set workerCellListCondensed replace-item 0
workerCellListCondensed [number] of turtle invadedWorkerCohortID ]
        if any? turtles with [ who = memoInvadedD ] [ set droneCellListCondensed replace-item 0
droneCellListCondensed [number] of turtle invadedDroneCohortID ]
      ]
    ]

    if KillAllMitesInCells2 = true
    [
      ask miteOrganisers
      [
        set workerCellListCondensed n-values (MAX_INVADED_MITES_WORKERCELL + 1) [ 0 ]
        set droneCellListCondensed n-values (MAX_INVADED_MITES_DRONECELL + 1) [ 0 ]
        let memoInvadedW invadedWorkerCohortID
        let memoInvadedD invadedDroneCohortID
        if any? turtles with [ who = memoInvadedW ] [ set workerCellListCondensed replace-item 0
workerCellListCondensed [number] of turtle invadedWorkerCohortID ]
        if any? turtles with [ who = memoInvadedD ] [ set droneCellListCondensed replace-item 0
droneCellListCondensed [number] of turtle invadedDroneCohortID ]
      ]
    ]

  ]

  ; removal drone brood:
  if (ContinuousBroodRemoval = true) or (DroneBroodRemoval = true and (day = RemovalDay1 or day
= RemovalDay2 or day = RemovalDay3 or day = RemovalDay4 or day = RemovalDay5))
  [
    ask dronePupaeCohorts
    [
      set number 0
      set number_healthy 0
      set number_infectedAsPupa 0
    ]
    ask miteOrganisers with [ age >= DRONE_PUPATION_AGE + 1 ]
    [
      set droneCellListCondensed n-values (MAX_INVADED_MITES_DRONECELL + 1) [ 0 ]
    ]
    CountingProc
  ]

  ; re-infestation of varroa-mites
  if AllowReinfestation = true
  [
    let additionalMites random-poisson MiteReinfestation
    if DailyForagingPeriod = 0 [ set additionalMites 0 ]
    if phoreticMites + additionalMites > 0
      [ set PhoreticMitesHealthyRate  (phoreticMites * phoreticMitesHealthyRate + additionalMites /
2) / (phoreticMites + additionalMites)] ; assumes 50% of new mites are infected with virus
    set PhoreticMites PhoreticMites + additionalMites
    set TotalMites TotalMites + additionalMites
```

```
 ]

 ask miteOrganisers  ; update the number of invaded mites for each mite organiser:
 [
   let counter 0
   set cohortInvadedMitesSum 0
   foreach workerCellListCondensed
   [
     set cohortInvadedMitesSum cohortInvadedMitesSum + (? * counter)
     set counter counter + 1
   ]
   set counter 0
   foreach droneCellListCondensed
   [
     set cohortInvadedMitesSum cohortInvadedMitesSum + (? * counter)
     set counter counter + 1
   ]
   set label cohortInvadedMitesSum
 ]


 ; end ***NEW FOR BEEHAVE_BEEMAPP2015***



end

;
**********************************************************************************
**********************************************************************************
************

;.....
.....
;           PLOT PROCEDURES
;.....
.....


;
**********************************************************************************
**********************************************************************************
************

to DoPlotsProc
; CAUTION: choosing "age forager squadrons", "mileometer" or "mean total km per day" will affect
; the sequence of random numbers!
; with-local-randomness [  ; to run the procedure is run without affecting subsequent random events
  if showAllPlots = true [ DrawForagingMapProc ]
```

```
 ask Signs with [ shape = "arrow" ]
 [
  facexy (xcor + 1000000) (ycor + (HoneyEnergyStore - HoneyEnergyStoreYesterday)
    / ( ENERGY_HONEY_per_g / 1000))

  set label  word "H: " precision ((HoneyEnergyStore - HoneyEnergyStoreYesterday)
    / ( ENERGY_HONEY_per_g * 1000 )) 2

  ifelse (HoneyEnergyStore - HoneyEnergyStoreYesterday)
   / ( ENERGY_HONEY_per_g * 1000 ) >= 0
   [ set color green ]
   [ set color red ]
 ]

 ask Signs with [ shape = "arrowpollen" ]
 [
  facexy (xcor - 100) (ycor + (PollenStore_g - PollenStore_g_Yesterday))
  set label word "P: " precision ((PollenStore_g - PollenStore_g_Yesterday) / 1000) 2
  ifelse (PollenStore_g - PollenStore_g_Yesterday) > 0
   [ set color green ]
   [ set color red ]
 ]

 ask Signs with [shape = "pete"]
 [
  ifelse VarroaTreatment = true
       or FeedBees = true
       or HoneyHarvesting = true
       or AddPollen
       or MergeWeakColonies = TRUE
   [ show-turtle]
   [ hide-turtle ]
 ]

 ; calling GenericPlottingProc (8x) with plotname & plotChoice as input:
 GenericPlottingProc "Generic plot 1" GenericPlot1
 GenericPlottingProc "Generic plot 2" GenericPlot2
 GenericPlottingProc "Generic plot 3" GenericPlot3
 GenericPlottingProc "Generic plot 4" GenericPlot4
 GenericPlottingProc "Generic plot 5" GenericPlot5
 GenericPlottingProc "Generic plot 6" GenericPlot6
 GenericPlottingProc "Generic plot 7" GenericPlot7
 GenericPlottingProc "Generic plot 8" GenericPlot8
; ] ; end "with-local-randomness"
end
;
******************************************************************************
******************************************************************************
************

to GenericPlotClearProc
```

```
 ; clear those plots, that only show output of 'today'

 let i 1
 while [ i <= N_GENERIC_PLOTS ]

 [
  let plotname (word "Generic plot " i)
   ; e.g. "Generic plot 1"
   if (i = 1 and (GenericPlot1 = "foragers today [%]" or GenericPlot1 = "active foragers today [%]"))
   or (i = 2 and (GenericPlot2 = "foragers today [%]" or GenericPlot2 = "active foragers today [%]"))
   or (i = 3 and (GenericPlot3 = "foragers today [%]" or GenericPlot3 = "active foragers today [%]"))
   or (i = 4 and (GenericPlot4 = "foragers today [%]" or GenericPlot4 = "active foragers today [%]"))
   or (i = 5 and (GenericPlot5 = "foragers today [%]" or GenericPlot5 = "active foragers today [%]"))
   or (i = 6 and (GenericPlot6 = "foragers today [%]" or GenericPlot6 = "active foragers today [%]"))
   or (i = 7 and (GenericPlot7 = "foragers today [%]" or GenericPlot7 = "active foragers today [%]"))
   or (i = 8 and (GenericPlot8 = "foragers today [%]" or GenericPlot8 = "active foragers today [%]"))
     [
      set-current-plot plotname
      clear-plot
     ]
  set i i + 1
 ]
end
;
**********************************************************************************
**********************************************************************************
************



to GenericPlottingProc [ plotname plotChoice ]
 set TotalEventsToday NectarFlightsToday + PollenFlightsToday + EmptyFlightsToday
 set-current-plot plotname

 set TotalWeightBees_kg
  ( TotalEggs * 0.0001     ; 0.0001g (wegg, HoPoMo)
   + TotalLarvae * 0.0457
      ; 0.0457g : average weight of a larva (using wlarva 1..5 from HoPoMo (p. 231)
   + TotalPupae * 0.16      ; 0.16g wpupa (HoPoMo)
   + (TotalIHbees + TotalForagers) * WEIGHT_WORKER_g      ; 0.1g wadult (HoPoMo)
   + TotalDroneEggs * 0.0001
   + TotalDrones * 0.22
      ; 0.22g (Rinderer, Collins, Pesante (1985), Apidologie)
   + TotalDroneLarvae *(0.1 * (0.22 / WEIGHT_WORKER_g))
     ; estimation of drone larva weight on basis of worker larva weight and
     ; adult worker:drone weight
     ; 0.10054 = 0.0457*2.2 = estimated drone larva weight
   + TotalDronePupae * (0.16 * (0.22 / WEIGHT_WORKER_g))
     ; estimation of drone pupa weight on basis of worker pupa weight and adult worker:drone
weight
  ) / 1000 ; [g] -> [kg]
```

```
 if plotChoice = "colony weight [kg]" ; total weight of the colony without hive/supers etc.
 [
   create-temporary-plot-pen "weight"
   plot TotalWeightBees_kg ;
 ]

 if plotChoice = "foragingPeriod"
   [
    create-temporary-plot-pen "period"
    plotxy ticks DailyForagingPeriod / 3600
   ]
 if plotChoice = "# completed foraging trips (E-3)"
   [
     create-temporary-plot-pen "# trips"
     plotxy ticks totalEventsToday / 1000
   ]

if plotChoice = "trips per hour sunshine (E-3)"
   [
     create-temporary-plot-pen "trips/h"
     ifelse DailyForagingPeriod > 0
      [ plotxy ticks (TotalEventsToday / 1000) / (DailyForagingPeriod / 3600) ]
      [ plotxy ticks 0 ]
   ]

if plotChoice = "active foragers [%]"
   [
     create-temporary-plot-pen "active%"
     set-plot-y-range  0 100
     set-plot-pen-mode 1 ; 1: bars
     ifelse TotalForagers > 0
       [ plotxy ticks (100 * SQUADRON_SIZE
             * (count foragersquadrons with [km_today > 0])) / TotalForagers ]
       [ plotxy ticks 0 ]
   ]

if plotChoice = "mean trip duration"
   [
     create-temporary-plot-pen "trip [min]"
     set-plot-pen-mode 1 ; 1: bars
     ifelse ForagingRounds > 0
      [ plotxy ticks ( DailyForagingPeriod  / (ForagingRounds * 60)) ]
       ; mean Foraging trip duration [min] on this day
      [ plotxy ticks 0 ] ; if no foraging takes place
   ]

if plotChoice = "mean total km per day"
   [
     create-temporary-plot-pen "km/d"
     set-plot-pen-mode 0 ; 0: lines
```

```
    ifelse count foragerSquadrons > 0
      [ plotxy ticks mean [km_today] of foragerSquadrons ]
      [ plotxy ticks 0 ]
  ]

if plotChoice = "mileometer"
  [
    create-temporary-plot-pen "km"
    set-plot-x-range  0 850
    set-plot-y-range  0 40
    set-plot-pen-mode 1 ; 1: bars
    set-plot-pen-interval 25
    histogram [ mileometer ] of foragerSquadrons
  ]


if plotChoice = "loads returning foragers [%]"
  [
    set totalEventsToday NectarFlightsToday + PollenFlightsToday + EmptyFlightsToday
    ifelse totalEventsToday > 0
      [
        create-temporary-plot-pen "nectar"
        set-plot-pen-color yellow
        plotxy ticks (100 * NectarFlightsToday) / totalEventsToday
        create-temporary-plot-pen "pollen"
        set-plot-pen-color orange
        plotxy ticks (100 * PollenFlightsToday) / totalEventsToday
        create-temporary-plot-pen "empty"
        set-plot-pen-color cyan
        plotxy ticks (100 * EmptyFlightsToday) / totalEventsToday
      ]
      [
        create-temporary-plot-pen "nectar"
        set-plot-pen-color yellow
        plotxy ticks 0
        create-temporary-plot-pen "pollen"
        set-plot-pen-color orange
        plotxy ticks 0
        create-temporary-plot-pen "empty"
        set-plot-pen-color cyan
        plotxy ticks 0
      ]
  ]

 if plotChoice = "broodcare [%]"
  [
    set-plot-y-range 0 150
    create-temporary-plot-pen "Protein"
      set-plot-pen-color orange
      plot ( ProteinFactorNurses * 100 )  ; Proteinfactor of nurses [%]
    create-temporary-plot-pen "Workload"
```

```
    if ((TotalIHbees + TotalForagers * FORAGER_NURSING_CONTRIBUTION)
      * MAX_BROOD_NURSE_RATIO) > 0 ; avoids division by 0
    [
      plot ( 100 * (TotalWorkerAndDroneBrood / ((TotalIHbees + TotalForagers
          * FORAGER_NURSING_CONTRIBUTION) * MAX_BROOD_NURSE_RATIO)) )
    ]

   create-temporary-plot-pen "Pollen"
     set-plot-pen-color green
     plot (PollenStore_g / IdealPollenStore_g) * 100
  ]


if plotChoice = "consumption [g/day]"
  [
   create-temporary-plot-pen "honey"
     set-plot-pen-color yellow
     plot (DailyHoneyConsumption / 1000)  ;[g/day]

   create-temporary-plot-pen "pollen"
     set-plot-pen-color orange
     plot (DailyPollenConsumption_g)   ;[g/day]
  ]


if plotChoice = "drones"
  [
   create-temporary-plot-pen "Eggs"  ; DRONE eggs
     set-plot-pen-color blue
     plot (TotalDroneEggs)
   create-temporary-plot-pen "Larvae"  ; DRONE larvae
     set-plot-pen-color yellow
     plot (TotalDroneLarvae)
   create-temporary-plot-pen "Pupae"   ; DRONE pupae
     set-plot-pen-color brown
     plot (TotalDronePupae)
   create-temporary-plot-pen "Drones"
     plot (TotalDrones)
  ]



if plotChoice = "colony structure workers"
  [
   create-temporary-plot-pen "Eggs"
     set-plot-pen-color blue
     plot (TotalEggs)
   create-temporary-plot-pen "Larvae"
     set-plot-pen-color yellow
     plot (TotalLarvae)
   create-temporary-plot-pen "Pupae"
```

```
      set-plot-pen-color brown
      plot (TotalPupae)
    create-temporary-plot-pen "IHbees"
      set-plot-pen-color orange
      plot (TotalIHbees)
    create-temporary-plot-pen "Foragers"
      set-plot-pen-color green
      plot (TotalForagers)
    create-temporary-plot-pen "Adults"
      set-plot-pen-color black
      plot (TotalForagers + TotalIHbees)
    create-temporary-plot-pen "Brood"
      set-plot-pen-color violet
      plot (TotalEggs + TotalLarvae + TotalPupae)
  ]



 let totalNectarAvailableToDay 0
 let totalPollenAvailableToDay 0
 ask flowerPatches
 [
   set totalNectarAvailableToDay totalNectarAvailableToDay + quantityMyl
   set totalPollenAvailableToDay totalPollenAvailableToDay + amountPollen_g
 ]

if plotChoice = "nectar availability [l]"
 [
   ifelse readInfile = false
     [
       create-temporary-plot-pen "Patch 0"
         set-plot-pen-color red
         plot (([ quantityMyl ] of flowerPatch 0 ) / 1000000 )  ;[l] nectar
       create-temporary-plot-pen "Patch 1"
         set-plot-pen-color green
         plot (([ quantityMyl ] of flowerPatch 1 ) / 1000000 )  ;[l] nectar
     ]
     [
       create-temporary-plot-pen "all patches"
       set-plot-pen-color yellow ; black
       plot (totalNectarAvailableToDay / 1000000 )  ;[l] nectar
     ]
 ]


 if plotChoice =  "pollen availability [kg]"
 [
   ifelse readInfile = false
     [
       create-temporary-plot-pen "Patch 0"
```

```
      set-plot-pen-color red
      plot (([ amountPollen_g ] of flowerPatch 0 ) / 1000 )  ; [kg] pollen
     create-temporary-plot-pen "Patch 1"
      set-plot-pen-color green
      plot (([ amountPollen_g ] of flowerPatch 1 ) / 1000 )  ; [kg] pollen
    ]
    [
     create-temporary-plot-pen "all patches"
      set-plot-pen-color orange; black
      plot (totalPollenAvailableToDay / 1000 )  ; [kg] pollen
    ]
 ]

 if plotChoice = "egg laying"
 [
  create-temporary-plot-pen "new eggs"
  plot (NewWorkerEggs)
 ]

  if plotChoice = "honey gain [kg]"
    [
     set-plot-y-range -3 10
     create-temporary-plot-pen "gain"
     set-plot-pen-mode 1 ; 1: bars
     ifelse (HoneyEnergyStore - HoneyEnergyStoreYesterday) / ( ENERGY_HONEY_per_g * 1000 ) < 0
      [ set-plot-pen-color red ]
      [ set-plot-pen-color black ]
     plotxy ticks (HoneyEnergyStore - HoneyEnergyStoreYesterday) / ( ENERGY_HONEY_per_g * 1000
)
    ]

  if plotChoice = "honey & pollen stores & hive [kg]"
  [ create-temporary-plot-pen "honey"
     set-plot-pen-color yellow
     plot (HoneyEnergyStore / ( ENERGY_HONEY_per_g * 1000 ) )  ;[ml] honey
  ; create-temporary-plot-pen "decent honey"
  ;    set-plot-pen-color brown
  ;   plot (TotalIHbees + TotalForagers ) * 0.0015
     ;; 1.5g honey per bee = estimated honey necessary for the colony to survive the winter
   create-temporary-plot-pen "pollen x 20"
     set-plot-pen-color orange
     plot 20 * (PollenStore_g / 1000)  ;[kg * 10] pollen stored in the colony in kg
  ]

  if plotChoice = "mites"
  [
   create-temporary-plot-pen "totalMites"
     plot (TotalMites)  ; # all mites (phoretic & in cells)
   create-temporary-plot-pen "phoreticMites"
     set-plot-pen-color brown
     plot (PhoreticMites)  ; # phoretic mites
```

```
  create-temporary-plot-pen "phoreticMitesInfected"
    set-plot-pen-color red
    plot (PhoreticMites * (1 - PhoreticMitesHealthyRate))  ; # infected phoretic mites
  create-temporary-plot-pen "phoreticMitesHealthy"
    set-plot-pen-color green
    plot (PhoreticMites * PhoreticMitesHealthyRate)  ; # healthy phoretic mites
  create-temporary-plot-pen "miteDrop x 10"
    set-plot-pen-color violet
    plot (DailyMiteFall * 10)  ; # dropping mites
]

if plotChoice = "proportion infected mites"
[
  create-temporary-plot-pen "proportion"
  ;if TotalMites > 0 [ plotxy ticks (1 - PhoreticMitesHealthyRate) ] ; ***NEW FOR
BEEHAVE_BEEMAPP2015***
   plotxy ticks (1 - PhoreticMitesHealthyRate)  ; ***NEW FOR BEEHAVE_BEEMAPP2015***
]

if plotChoice = "aff & lifespan"
[
  create-temporary-plot-pen "aff"
    set-plot-y-range 0 200
    set-plot-pen-mode 1 ; 1: bars
    if count foragerSquadrons with [age = aff] > 0
      [ plotxy ticks (aff) ]
  create-temporary-plot-pen "lifespan"
    set-plot-pen-color green
    set-plot-pen-mode 2 ; 2: dots
    ifelse (DeathsAdultWorkers_t > 0)
      and ((SumLifeSpanAdultWorkers_t / deathsAdultWorkers_t) < MIN_AFF)
        [ plot-pen-down ]
        [ plot-pen-up  ]
    plot (SumLifeSpanAdultWorkers_t / (DeathsAdultWorkers_t + 0.0000001)) ; to avoid division by 0
]

if plotChoice =  "age forager squadrons"
[
  set-plot-y-range 0 10
  set-plot-x-range 0 300

  create-temporary-plot-pen "foragersHealthy"
    set-plot-pen-mode 1 ; 1: bars
    set-plot-pen-interval 1
    histogram [ age ] of foragerSquadrons
      with [ infectionState = "healthy" ]

  create-temporary-plot-pen "foragersDiseased"
    set-plot-pen-mode 1 ; 1: bars
    set-plot-pen-interval 1
    set-plot-pen-color red
```

```
    histogram [ age ] of foragerSquadrons
      with [ infectionState = "infectedAsPupa" ]
      ; infectedAsPupa = true or infectedAsAdult = true ]

   create-temporary-plot-pen "foragersCarrier"
    set-plot-pen-mode 1 ; 1: bars
    set-plot-pen-interval 1
    set-plot-pen-color blue
    histogram [ age ] of foragerSquadrons
      with [ infectionState = "infectedAsAdult" ]
 ]

end


;
******************************************************************************
******************************************************************************
************

to DrawForagingMapProc
; CAUTION: choice of ForagingMap and DotDensity affects the sequence of random numbers!
; with-local-randomness [  ; procedure is run without affecting subsequent random events
 set-current-plot "foraging map"
 set-current-plot-pen "default"
 clear-plot
 let xplot 0
 let yplot 0
 ask flowerPatches
 [
  if ForagingMap = "Nectar foraging"
  [
   repeat nectarVisitsToday * DotDensity
   [
    let radius sqrt(size_sqm / pi)
     ; the (hypothetical) radius of the patch (assumed to be circular)

    set xplot (xcorMap - radius) + (random-float (2 * radius))
     ; x coordinate randomly chosen from centre +- radius

    let yRange sqrt((radius ^ 2) - ((xplot - xcorMap) ^ 2))
     ; calculate the range of possible y-coordinates for chosen x-coordinate,

    set yplot (ycorMap - yRange) + (random-float (2 * yRange))
     ; y coordinate randomly chosen from the range of possible values

    set-plot-pen-color yellow
    plotxy xplot yplot
   ]
  ]
```

```
if ForagingMap = "Pollen foraging"
[
  repeat pollenVisitsToday * DotDensity
  [
    let radius sqrt(size_sqm / pi)
    ; the (hypothetical) radius of the patch (assumed to be circular)

    set xplot (xcorMap - radius) + (random-float (2 * radius))
      ; x coordinate randomly chosen from centre +- radius

    let yRange sqrt((radius ^ 2) - ((xplot - xcorMap) ^ 2))
      ; calculate the range of possible y-coordinates for chosen x-coordinate,

    set yplot (ycorMap - yRange) + (random-float (2 * yRange))
      ; y coordinate randomly chosen from the range of possible values )

    set-plot-pen-color orange
    plotxy xplot yplot
  ]
]

if ForagingMap = "All visits"
[
  repeat (nectarVisitsToday + pollenVisitsToday) * DotDensity
  [
    let radius sqrt(size_sqm / pi)
      ; the (hypothetical) radius of the patch (assumed to be circular)

    set xplot (xcorMap - radius) + (random-float (2 * radius))
      ; x coordinate randomly chosen from centre +- radius

    let yRange sqrt((radius ^ 2) - ((xplot - xcorMap) ^ 2))
      ; calculate the range of possible y-coordinates for chosen x-coordinate,

    set yplot (ycorMap - yRange) + (random-float (2 * yRange))
      ; y coordinate randomly chosen from the range of possible values

    set-plot-pen-color black
    plotxy xplot yplot
  ]
]

 if ForagingMap = "All patches"
 [
   repeat 10000 * DotDensity
   [
     let radius sqrt(size_sqm / pi)
       ; the (hypothetical) radius of the patch (assumed to be circular)

     set xplot (xcorMap - radius) + (random-float (2 * radius))
       ; x coordinate randomly chosen from centre +- radius
```

```
    let yRange sqrt((radius ^ 2) - ((xplot - xcorMap) ^ 2))
      ; calculate the range of possible y-coordinates for chosen x-coordinate,

    set yplot (ycorMap - yRange) + (random-float (2 * yRange))
      ; y coordinate randomly chosen from the range of possible values

    if patchType = "YellowField"
       or patchType = "OilSeedRape"
    [
      set-plot-pen-color yellow
    ]
    if patchType = "RedField" [ set-plot-pen-color red ]
    if patchType = "BlueField" [ set-plot-pen-color blue ]
    if patchType = "GreenField" [ set-plot-pen-color green ]
    plotxy xplot yplot
  ]
]

if ForagingMap = "Available patches"
[
  let proportionPollen 0
  let pollenAvailable amountPollen_g / POLLENLOAD
    ; # pollen loads available

  let nectarAvailable quantityMyl / CROPVOLUME
    ; # crop loads available

  if pollenAvailable + nectarAvailable > 0
  [
    set proportionPollen pollenAvailable / (pollenAvailable + nectarAvailable)
  ]

  repeat round sqrt((pollenAvailable + nectarAvailable) * DotDensity)
    ; sqrt to avoid too many repeats
  [
    let radius sqrt(size_sqm / pi)
      ; the (hypothetical) radius of the patch (assumed to be circular)

    set xplot (xcorMap - radius) + (random-float (2 * radius))
      ; x coordinate randomly chosen from centre +- radius

    let yRange sqrt((radius ^ 2) - ((xplot - xcorMap) ^ 2))
      ; calculate the range of possible y-coordinates for chosen x-coordinate,

    set yplot (ycorMap - yRange) + (random-float (2 * yRange))
      ; y coordinate randomly chosen from the range of possible values

    ifelse random-float 1 < proportionPollen
      [ set-plot-pen-color orange ]
      [ set-plot-pen-color yellow ]
```

```
      plotxy xplot yplot
    ]
   ]

   if ForagingMap = "Nectar and Pollen"
   [
    let proportionPollen 0
    if pollenVisitsToday + nectarVisitsToday > 0
    [
     set proportionPollen pollenVisitsToday
      / ( pollenVisitsToday
       + nectarVisitsToday )
    ]

    repeat (pollenVisitsToday + nectarVisitsToday) * DotDensity
    [
     let radius sqrt(size_sqm / pi)
      ; the (hypothetical) radius of the patch (assumed to be circular)

     set xplot (xcorMap - radius) + (random-float (2 * radius))
      ; x coordinate randomly chosen from centre +- radius

     let yRange sqrt((radius ^ 2) - ((xplot - xcorMap) ^ 2))
     set yplot (ycorMap - yRange) + (random-float (2 * yRange))
     ifelse random-float 1 < proportionPollen
      [ set-plot-pen-color orange ]
      [ set-plot-pen-color yellow ]
     plotxy xplot yplot
    ]
   ]
  ] ; end of: "Ask flowerpatches"

 set-plot-pen-color brown  ; draw the colony:
 repeat 10000
 [
  plotxy (-50 + random 100) (-50 + random 100)
 ]
; ]  ; end "local randomness"
end

;
*******************************************************************************
*******************************************************************************
************

to WriteToFileProc
 ; writes data in file, copied from: Netlogo: Library:
 ; Code Examples: "File Output Example"

 let year ceiling (ticks / 365)
 foreach sort flowerPatches
```

```
 [
  ask ?
  [
    file-print
      ( word year " " word ticks " " ForagingRounds " " word self
        " distance: " distanceToColony
        " concentration: " nectarConcFlowerPatch
        " EEF: " EEF
        " quantity: " quantityMyl)
  ]
 ]

 foreach sort foragerSquadrons
 [
  ask ?
  [
    file-print
     (word year " " word ticks " " ForagingRounds " "  word self
       " age: " age
       " km: " mileometer)
  ]
 ]

end

;
*******************************************************************************
*******************************************************************************
************

to-report DateREP
  let month-names (list "January" "February" "March" "April" "May" "June" "July" "August"
"September" "October" "November" "December")
  let days-in-months (list 31 28 31 30 31 30 31 31 30 31 30 31)


  let year floor (ticks / 365.01) + 1
  let month 0
  let dayOfYear remainder ticks 365
  if dayOfYear = 0 [ set dayOfYear 365 ]
  let dayOfMonth 0
  let sumDaysInMonths 0
  while [ sumDaysInMonths < dayOfYear ]
  [
    set month month + 1
    set sumDaysInMonths sumDaysInMonths + item (month - 1) days-in-months
    set dayOfMonth dayOfYear - sumDaysInMonths + item (month - 1) days-in-months
  ]

  report (word dayOfMonth "  " (item (month - 1) month-names) " " year )
```

```
end

;
*******************************************************************************
*******************************************************************************
************

to ReadFileProc
  ; reads data in from file, copied from: Netlogo: Library:
  ; Code Examples: "File Input Example"

  ifelse ( file-exists? INPUT_FILE )
    ; We check to make sure the file exists first
    [
      set AllDaysAllPatchesList []
        ; IF: data are saved in a list (list still empty)

      file-open INPUT_FILE
      let dustbin file-read-line
        ; first line of input file with headings is read - but not used for anything

      while [ not file-at-end? ]
      [
        set AllDaysAllPatchesList sentence AllDaysAllPatchesList
          (list (list file-read file-read file-read file-read file-read
                      file-read file-read file-read file-read file-read
                      file-read file-read file-read file-read file-read))]
          ; 15 data colums are read in
        file-close  ; closes file
        set N_FLOWERPATCHES ((length AllDaysAllPatchesList) / 365)
        if (N_FLOWERPATCHES mod 1) != 0
        [
          user-message "Error in Infile - wrong number of lines"
          set BugAlarm true
        ]
    ] ; end "ifelse"
    [
      user-message "There is no such ~~file~~INPUT_FILE in current directory!"
    ]
end

;
*******************************************************************************
*******************************************************************************
************


to ReadBeeMappFileProc
  ; reads colony data in from file, created by the BeeMapp app

  ifelse ( file-exists? BeeMapp_FILE )
```

```
  [
    set AllBeeMappCorrectionsList []
    file-open BeeMapp_FILE
    let dustbin file-read-line
      ; first line of input file with headings is read - but not used for anything

    while [ not file-at-end? ]
    [
      set AllBeeMappCorrectionsList sentence AllBeeMappCorrectionsList ; 10 columns in BeeMapp
input file:
        (list (list ; repeat nColumns [ file-read ]
                  file-read file-read file-read file-read
                  file-read file-read file-read file-read
             ))]
      set AssessmentNumber 0
      ;(list (list file-read-line ))]


      file-close
  ] ; end "ifelse"
  [
    user-message "There is no such BeeMapp_FILE in current directory!"
  ]
end

;
*****************************************************************************************
*****************************************************************************************
************

to BeeMappCorrectionProc ; ***NEW FOR BEEHAVE_BEEMAPP2015***

 let nextBeeMappCorrectionList item AssessmentNumber AllBeeMappCorrectionsList

 if ticks = item 1 nextBeeMappCorrectionList  ; if day = date of colony next colony assessment
 [
  ; correct honey stores according to real honey stores:
  set HoneyEnergyStore ENERGY_HONEY_per_g * 1000 * item 7 nextBeeMappCorrectionList ;

  ; correct number of workers according to real colony size:
  let correctedNumberWorkers item 6 nextBeeMappCorrectionList ;
  if correctedNumberWorkers < 0 [ set correctedNumberWorkers 0 ]

  ; correct # foragers:
  let correctedNumberForagers correctedNumberWorkers * (totalForagers / (totalIHbees +
totalForagers)) ;
  let correctedNumberForagerSquadrons round (correctedNumberForagers / SQUADRON_SIZE)

  ifelse correctedNumberForagerSquadrons * SQUADRON_SIZE < totalForagers
  [
```

```
    repeat totalForagers / SQUADRON_SIZE - correctedNumberForagerSquadrons   ; if foragers have to be REMOVED from the simulation
      [ ask one-of foragerSquadrons [ die ] ]
  ]
  [
    repeat correctedNumberForagerSquadrons - totalForagers / SQUADRON_SIZE    ; if foragers have to be ADDED to the simulation
      [ ask one-of foragerSquadrons [ hatch 1 ] ]
  ]

  ; correct # in-hive bees:
  let correctedNumberIHbees correctedNumberWorkers - correctedNumberForagerSquadrons * SQUADRON_SIZE
  let changeNumberBy1 0

  ifelse correctedNumberIHbees - totalIHbees < 0
  [ set changeNumberBy1 -1 ]  ; if IHbees have to be REMOVED from the simulation
  [ set changeNumberBy1 1 ]   ; if IHbees have to be ADDED to the simulation

  repeat sqrt ((correctedNumberIHbees - totalIHbees) ^ 2)
  [
    let applyToCohorts IHbeecohorts
    if count IHbeecohorts with [ number > 0 ] > 0
      [ set applyToCohorts IHbeecohorts with [ number > 0 ]  ]

    ;ask one-of IHbeecohorts with [ number > 0 ]
    ask one-of applyToCohorts

    [
      let chooseBee 1 + random number  ; to determine which sub-cohort (healthy, infected as pupa or as adult) is affected, depending on number of bees in each cohort
      let changeHealthy false
      let changeInfPupa false
      let changeInfAdult false

      ; determine which sub-cohort is changed:
      if chooseBee <= number_healthy
        [ set changeHealthy true ]

      if chooseBee > number_healthy and chooseBee <= number_healthy + number_infectedAsPupa
        [ set changeInfPupa true ]

      if chooseBee > number_healthy + number_infectedAsPupa and chooseBee <= number_healthy + number_infectedAsPupa + number_infectedAsAdult
        [ set number_infectedAsAdult number_infectedAsAdult + changeNumberBy1 ]


      ; do the change in numbers (separate step, otherwise errors might occur)
      set number number + changeNumberBy1

      if changeHealthy = true
```

```
        [ set number_healthy number_healthy + changeNumberBy1 ]

      if changeInfPupa = true
        [ set number_infectedAsPupa number_infectedAsPupa + changeNumberBy1 ]

      if changeInfAdult = true
        [ set number_infectedAsAdult number_infectedAsAdult + changeNumberBy1 ]
    ]
  ]


; NEW for new BeeMapp version that allows assessment of number of capped brood cells (31.03.2016):
; correct # pupae:
  let correctedNumberPupae item 5 nextBeeMappCorrectionList
  if correctedNumberPupae > 1
  [
    set changeNumberBy1 0
    ifelse correctedNumberPupae - (TotalPupae + TotalDronePupae) < 0
    [ set changeNumberBy1 -1 ]  ; if IHbees have to be REMOVED from the simulation
    [ set changeNumberBy1 1 ]   ; if IHbees have to be ADDED to the simulation
    repeat sqrt ((correctedNumberPupae - (TotalPupae + TotalDronePupae)) ^ 2)
    [
      let applyToCohorts pupaeCohorts  ; defines the pupal cohorts that are affected, here: all worker pupae (only if no worker pupae at all are currently present in the model
      if count pupaeCohorts with [ number > 0 ] > 0          ; otherwise (i.e. IF there are worker pupae present), then only those worker pupae cohorts with number > 0 are affected. Hence gaps (e.g. due to lack of pollen) still remain
        [ set applyToCohorts pupaeCohorts with [ number > 0 ]  ]
      if random (TotalPupae + TotalDronePupae) > TotalPupae ; in this case, the number of drone pupae will be modified
      [
        set applyToCohorts dronePupaeCohorts
        if count dronePupaeCohorts with [ number > 0 ] > 0
        [ set applyToCohorts pupaeCohorts with [ number > 0 ]  ]

      ]
      ask one-of applyToCohorts
      [
        set number number + changeNumberBy1
        set number_healthy number_healthy + changeNumberBy1
        set TotalPupae TotalPupae + changeNumberBy1
      ]
    ]
  ]


  ; PRESENCE/ABSENCE of QEL: -1: not assessed, 0: not present, 1: present
  ; new queen, if no queen was found in real colony
  if item 2 nextBeeMappCorrectionList = 0
    [ set Queenage 0 ]
```

```
  ; remove eggs, if no eggs were found in real colony
  if item 3 nextBeeMappCorrectionList = 0
  [
    ask eggcohorts [ set number 0 ]
    set NewWorkerLarvae 0
    ask droneeggcohorts [ set number 0]
    set NewDroneLarvae 0
  ]

  ; remove larvae, if no larvae were found in real colony
  if item 4 nextBeeMappCorrectionList = 0
  [
    ask larvaecohorts [ set number 0 ]
    set NewWorkerPupae 0
    ask dronelarvaecohorts [ set number 0 ]
    set NewDronePupae 0
  ]

  ; remove pupae, if no pupae were found in real colony
  if item 5 nextBeeMappCorrectionList = 0
  [
    ask pupaecohorts [ set number 0 set number_healthy 0 set number_infectedAsPupa 0 ]
    set NewIHbees 0
    set NewIHbees_healthy 0
    ask dronepupaecohorts [ set number 0 set number_healthy 0 set number_infectedAsPupa 0 ]
    set NewDrones 0
    set NewDrones_healthy 0
  ]

  if nextBeeMappCorrectionList != last AllBeeMappCorrectionsList ; if current correction is last item
in file/AllBeeMappCorrectionsList, then AssessmentNumber is no longer increased
  [
    set AssessmentNumber AssessmentNumber + 1
  ]
 ]

 CountingProc

end


;
**********************************************************************************************************************************
**********

to DefaultProc
; new variables:
set AllowReinfestation FALSE
```

```
;set BeeMapp_FILE "ColonyAssessment.txt"
set ContinuousBroodRemoval FALSE
set DroneBroodRemoval FALSE
set EfficiencyPhoretic2 0
; FrameType: no default setting
; HiveType: no default setting
set KillAllMitesInCells FALSE
set KillAllMitesInCells2 FALSE
set KillOpenBrood FALSE
set KillOpenBrood2 FALSE
set MiteReinfestation 0.1
set ReadBeeMappFile FALSE
set RemovalDay1 100
set RemovalDay2 140
set RemovalDay3 180
set RemovalDay4 220
set RemovalDay5 240
set TreatmentDay2 0
set TreatmentDuration2 0
; WeatherFile: no default setting


; new on interface (unchanged default values):
set EfficiencyPhoretic 0.115
set TreatmentDay 270   ; 270: 27.September
set TreatmentDuration 40  ; (28-40d) Fries et al. 1994
set AddedPollen_kg 0.5


; old variables, new default values:
set GenericPlot1 "honey & pollen stores [kg]"

; old variables, removed:
; set Testing "SIMULATION - NO TEST"
; old & unchanged (Beehave2013):
set AddPollen FALSE
set AlwaysDance  FALSE
set CONC_G  1.5
set CONC_R 1.5
set ConstantHandlingTime  FALSE
set CRITICAL_COLONY_SIZE_WINTER  4000
set Details  TRUE
set DANCE_INTERCEPT 0 ; -17.7
set DANCE_SLOPE 1.16
set DETECT_PROB_G 0.2
set DETECT_PROB_R 0.2
set DISTANCE_G  500
set DISTANCE_R 1500
set DotDensity  0.01 ; (affects sequence of random numbers)
set EggLaying_IH  TRUE
set Experiment  "none"
```

```
set FeedBees FALSE
set ForagingMap "Nectar and Pollen" ; (affects sequence of random numbers)
set GenericPlot2 "colony structure workers"
set GenericPlot3 "broodcare [%]"
set GenericPlot4 "mites"
set GenericPlot5 "nectar availability [l]"
set GenericPlot6 "pollen availability [kg]"
set GenericPlot7 "mean trip duration"
set GenericPlot8 "foragers today [%]"
set HarvestingDay 135
set HarvestingPeriod 80
set HarvestingTH 20
set HoneyHarvesting_FALSE
set HoneyIdeal_FALSE
;set INPUT_FILE_"Input_2-1_FoodFlow.txt"
set MAX_BROODCELLS_2000099
set MAX_km_PER_DAY_7299
set MAX_HONEY_STORE_kg_50
set MergeColoniesTH 5000
set MergeWeakColonies FALSE
set MiteReproductionModel_"Martin"
set ModelledInsteadCalcDetectProb_FALSE
set N_INITIAL_BEES_10000
set N_INITIAL_MITES_HEALTHY_0
set N_INITIAL_MITES_INFECTED_0
set POLLEN_G_kg 1.0
set POLLEN_R_kg 1.0
set PollenIdeal_FALSE
set ProbLazinessWinterbees 0 ; 0.7
set QUANTITY_G_l 20
set QUANTITY_R_l 20
set QueenAgeing_FALSE
; RAND_SEED: no default setting
set ReadInfile false
set RemainingHoney_kg_5
set SeasonalFoodFlow TRUE
set SHIFT_G -40
set SHIFT_R 30
set ShowAllPlots_TRUE
set SQUADRON_SIZE_100
set StopDead_TRUE
set Swarming_"No swarming"
set TIME_NECTAR_GATHERING 1200
set TIME_POLLEN_GATHERING 600
set VarroaTreatment_FALSE
set Virus_"DWV"
set Weather "Rothamsted (2009)" ; "Rothamsted (2009-2011)"
set WriteFile_FALSE
;set X_Days_7
```

```
end

;
*****************************************************************************
*****************************************************************************
************
;*** END ******** END ******** END ******** END ******** END ********
END ******** END ******** END ******** END ******** END **
;
*****************************************************************************
*****************************************************************************
************
```