```
__includes [ "parameters.nls" "reds.nls" "greens.nls" ]

; definition des especes
breed [ Explorers Explorer ]
breed [ RocketLaunchers RocketLauncher ]
breed [ Harvesters Harvester ]
breed [ Bases Base ]
breed [ Missiles Missile ]
breed [ Fafs Faf ]
breed [ Burgers Burger ]
breed [ Seeds Seed ]
breed [ Walls Wall ]
breed [ Perceptions Perception ]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;; definition des variables
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; les variables communes à tous les robots
;
turtles-own [ energy                 ;; l'energie de l'agent
              carrying-food?         ;; la quantite de nourriture transportee
              detection-range        ;; le rayon de perception
              speed                  ;; la vitesse
              fd-ok?                 ;; l'agent a-t-il le droit de se deplacer ? (un seul deplacement par tour)
              percept                ;; un agent pour visualiser la zone de perception
              my-bases               ;; les bases de l'agent
              mem0 mem1 mem2         ;; six zones memoires a utiliser pour la programmation des agents
              mem3 mem4 mem5         ;; chacune peut contenir un scalaire ou une liste de 2 valeurs
              friend                 ;; la couleur de mon équipe
              ennemy                 ;; la couleur de l'ennemi
              death-burgers          ;; le nombre de burgers relâchés à la mort d'un agent
            ]
;
; les variables specifiques aux rocket launchers
;
RocketLaunchers-own [ nb-missiles    ;; le nb de missiles transportes
                      max-missiles   ;; le nb max de missiles autorise
                      nb-fafs        ;; le nb de missiles de type "fire and forget"
                      max-fafs       ;; le nb max de missiles de type "fire and forget" autorise
                      waiting        ;; une tempo entre 2 tirs
                    ]

;
; les variables specifiques aux harvesters
;
Harvesters-own [ carried-walls-nrj    ;; l'énergie des murs transportés
               ]

;
; les variables specifiques aux bases
;
Bases-own [ nb-missiles              ;; le nb de missiles transportes
            max-missiles             ;; le nb max de missiles autorise
            nb-fafs                  ;; le nb de missiles de type "fire and forget"
            max-fafs                 ;; le nb max de missiles de type "fire and forget" autorise
            waiting                  ;; une tempo entre 2 tirs
            mem6 mem7 mem8           ;; la mémoire de la base est supérieure à celle des autres robots
            mem9 mem10 mem11         ;; chaque emplacement mémoire peut contenir une liste
            create-ok?
          ]
;
; les variables specifiques aux missiles
;
Missiles-own [ my-range ]            ;; la portee du missile
;
; les variables specifiques aux fafs
;
Fafs-own [ my-range                  ;; la portee du faf
           target                    ;; la cible du faf
         ]
;
```

```
; les variables specifiques aux perceptions (= cercles centres sur les robots pour visualiser leur sphère de
perception)
;
Perceptions-own [ agt my-range ]
;
; les variables specifiques aux graines
;
Seeds-own [ age ]
Burgers-own [ burger-breed          ;; l'espèce de l'agent qui l'a créé
              who-id                ;; l'identifiant du robot qui l'a créé
            ]

;
; les variables globales
;
globals [ victoire                    ;; 1 si victoire des verts / 2 si victoire des rouges
          duree                       ;; la duree d'une partie
          energy_green                ;; la quantite d'energie de l'equipe verte
          energy_red                  ;; la quantite d'energie de l'equipe rouge

          wall-cost                   ;; le cout de fabrication d'un mur
          wall-nrj                    ;; l'energie initiale d'un mur
          seed-cost                   ;; le cout d'une graine
          max-seeds                   ;; la quantite max de graines par patch
          maturation-time             ;; le temps de maturation des graines
          burger-decay                ;; la vitesse de pourrissement des burgers

          base-nrj                    ;; l'energie d'une base toute neuve
          base-perception             ;; la distance de perception d'une base
          base-speed                  ;; la vitesse d'une base
          base-nb-missiles            ;; le nombre initial de missiles
          base-max-missiles           ;; le nombre maximal de missiles
          base-nb-fafs                ;; le nombre initial de fafs
          base-max-fafs               ;; le nombre maximal de fafs
          base-waiting                ;; la temporisation entre 2 envois de missiles
          base-burgers                ;; le nb de burgers créés en cas de mort d'une base

          rocket-launcher-cost        ;; le cout de fabrication d'un rocket-launcher
          rocket-launcher-nrj         ;; l'energie d'un rocket-launcher tout neuf
          rocket-launcher-perception  ;; la distance de perception d'un rocket-launcher
          rocket-launcher-speed       ;; la vitesse d'un rocket-launcher
          rocket-launcher-metabolism  ;; le metabolisme d'un rocket-launcher (consommation d'energie a chaque
tour)
          rocket-launcher-nb-missiles ;; le nombre initial de missiles
          rocket-launcher-max-missiles ;; le nombre maximal de missiles
          rocket-launcher-nb-fafs     ;; le nombre initial de fafs
          rocket-launcher-max-fafs    ;; le nombre maximal de fafs
          rocket-launcher-waiting     ;; la temporisation entre 2 envois de missiles
          rocket-launcher-burgers     ;; le nb de burgers créés en cas de mort d'un rocket-launcher

          explorer-cost               ;; le cout de fabrication d'un explorer
          explorer-nrj                ;; l'energie d'un explorer tout neuf
          explorer-perception         ;; la distance de perception d'un explorer
          explorer-speed              ;; la vitesse d'un explorer
          explorer-metabolism         ;; le metabolisme d'un explorer (consommation d'energie a chaque tour)
          explorer-burgers            ;; le nb de burgers créés en cas de mort d'un explorer

          harvester-cost              ;; le cout de fabrication d'un harvester
          harvester-nrj               ;; l'energie d'un harvester tout neuf
          harvester-perception        ;; la distance de perception d'un harvester
          harvester-speed             ;; la vitesse d'un harvester
          harvester-metabolism        ;; le metabolisme d'un harvester (consommation d'energie a chaque tour)
          harvester-burgers           ;; le nb de burgers créés en cas de mort d'un harvester

          missile-cost                ;; le cout de creation d'un missile
          missile-range               ;; la portee d'un missile
          missile-robot-damage        ;; les dommages d'un missile sur un robot
          missile-base-damage         ;; les dommages d'un missile sur une base
          missile-speed               ;; la vitesse d'un missile

          faf-cost                    ;; le cout de creation d'un faf
          faf-range                   ;; la portee d'un faf
```

```
          faf-robot-damage            ;; les dommages d'un faf sur un robot
          faf-base-damage             ;; les dommages d'un faf sur une base
          faf-speed                   ;; la vitesse d'un faf

          collision-damage            ;; dégâts en cas de collision

          burger-periodicity          ;; la periodicite avec laquelle les burgers sont ajoutes
          burger-quantity             ;; le nb de burgers crees a chaque fois
          wild-burger-min-nrj         ;; l'energie minimale d'un burger sauvage
          wild-burger-max-nrj         ;; l'energie maximale d'un burger sauvage
          seeded-burger-min-nrj       ;; l'energie minimale d'un burger cultive
          seeded-burger-max-nrj       ;; l'energie maximale d'un burger cultive
]

to test-int
  let i 2000000000
  show i
  while [i > 0] [set i i + 1]
  show i
end


;
; Procedure d'initialisation
;
to setup [ config ]
  ; efface tout
  clear-all
  init-parameters

  ; paramètres generaux
  set victoire 0
  ; cree 2 bases de chaque couleur
  new-Base red green 30 10
  new-Base red green 30 -10
  new-Base green red -30 10
  new-Base green red -30 -10

  ; demande aux bases de creer des robots
  ask Bases with [color = green] [
    set my-bases Bases with [ color = green ]
    ; appelle la procedure d'activation correspondant a la couleur
    initGreenBase
  ]
  ask Bases with [color = red] [
    set my-bases Bases with [ color = red ]
    ; appelle la procedure d'activation correspondant a la couleur
    initRedBase
  ]

  ; cree des obstacles
  new-random-walls

  ; ajoute de la nourriture
  repeat 3 [ new-random-burgers burger-quantity ]

  ; met à jour l'affichage des ressources globales des 2 equipes
  update_energy_watches

  reset-ticks
end

;
; Procedure principale
;
to go
  ; reinitialise la capacité de se déplacer
  ask turtles [ set fd-ok? true ]
  ; réinitialise la capacité à créer des robots ou des murs pour les bases
  ask Bases [ set create-ok? true ]
  ; si plus de bases vertes ni bleues, victoire des rouges
  if not any? Bases with [ color = green ][ print "Victoire des rouges" set victoire 1 stop ]
```

```
; si plus de bases rouges ni bleues, victoire des verts
if not any? Bases with [ color = red ][ print "Victoire des verts" set victoire 2 stop ]

; pour tous les explorers
ask Explorers [
  ; decremente l'energie
  set energy energy - explorer-metabolism
  ; teste s'ils sont toujours en vie
  mort
  ; affiche ou non un label sur l'agent
  display-label
  ; appelle la procedure d'activation correspondant a la couleur
  ifelse (color = green)
  [ carefully[goGreenExplorer][show error-message] ]
  [ carefully[goRedExplorer][show error-message] ]
]

; pour tous les rocket-launchers
ask RocketLaunchers [
  ; decremente l'energie
  set energy energy - rocket-launcher-metabolism
  ; teste s'ils sont toujours en vie
  mort
  ; decremente le delai d'attente avant de pouvoir lancer un nouveau missile
  if (waiting > 0) [ set waiting waiting - 1 ]
  ; affiche ou non un label sur l'agent
  display-label
  ; appelle la procedure d'activation correspondant a la couleur
  ifelse (color = green)
  [ carefully[goGreenRocketLauncher][show error-message] ]
  [ carefully[goRedRocketLauncher][show error-message] ]
]

; pour tous les moissonneurs
ask Harvesters [
  ; decremente l'energie
  set energy energy - harvester-metabolism
  ; teste s'ils sont toujours en vie
  mort
  ; affiche ou non un label sur l'agent
  display-label
  ; appelle la procedure d'activation correspondant a la couleur
  ifelse (color = green)
  [ carefully[goGreenHarvester][show error-message] ]
  [ carefully[goRedHarvester][show error-message] ]
]

; pour toutes les bases
ask Bases [
  ; teste si elles sont toujours en vie
  mort
  ; decremente le delai d'attente avant de pouvoir lancer un nouveau missile
  if (waiting > 0) [ set waiting waiting - 1 ]
  ; convertit la nourriture recuperee en energie
  convert-food-into-energy
  ; affiche ou non un label sur l'agent
  display-label
  ; appelle la procedure d'activation correspondant a la couleur
  ifelse (color = green)
  [ carefully[goGreenBase][show error-message] ]
  [ carefully[goRedBase][show error-message] ]
]

; guidage des missiles
ask Missiles [ carefully[go-missile][show error-message] ]

; guidage des missiles
ask Fafs [ carefully[go-faf][show error-message] ]

; pousse des graines
ask Seeds [ carefully[grow-seed][show error-message] ]
; pourrissement des burgers
```

```
  ask Burgers [ set energy energy - burger-decay if (energy <= 0) [die] ]

  ; affichage des spheres de perception
  ask Perceptions [ carefully[go-perception][show error-message] ]

  ask Walls [ if (energy <= 0) [die] ]

  ; ajout aleatoire de nouveaux burgers
  if (random burger-periodicity = 0) [ carefully[new-random-burgers burger-quantity][show error-message] ]


  ; met à jour l'affichage des ressources globales des 2 equipes
  update_energy_watches
  tick
  if (ticks = duree) [ stop ]
end

to-report compute-energy [ c ]
  report round (sum [energy] of Bases with [ color = c ] +
  sum [energy] of Explorers with [ color = c ] +
  sum [energy] of Harvesters with [ color = c ] +
  sum [carrying-food?] of Harvesters with [ color = c ] +
  sum [energy] of RocketLaunchers with [ color = c ] +
  sum [nb-missiles * missile-cost] of RocketLaunchers with [ color = c ] +
  sum [nb-fafs * faf-cost] of RocketLaunchers with [ color = c ] +
  sum [nb-missiles * missile-cost] of bases with [ color = c ] +
  sum [nb-fafs * faf-cost] of Bases with [ color = c ])
end

to update_energy_watches
  set energy_red compute-energy red
  set energy_green compute-energy green
end

;
; Creation d'une nouvelle base de couleur 'c' a la position ('x', 'y')
;
to new-base [ c en x y ]
  ; cree la base
  create-Bases 1 [
    ; initialise la taille, la couleur, la position
    set size 2
    set color c
    set friend c
    set ennemy en
    set xcor x
    set ycor y
    ; initialise l'energie
    set energy base-nrj
    ; initialise le rayon de perception
    set detection-range base-perception
    ; les bases ne bougent pas
    set speed 0
    set label ""
    set mem0 0
    set mem1 0
    set mem2 0
    set mem3 0
    set mem4 0
    set mem5 0
    set mem6 0
    set mem7 0
    set mem8 0
    set mem9 0
    set mem10 0
    set mem11 0
    ; la base a le droit de créer un robot par tour
    set create-ok? true
    ; initialise le nb de missiles
    set nb-missiles base-nb-missiles
    set max-missiles base-max-missiles
    ; initialise le nb de faf
```

```
    set nb-fafs base-nb-fafs
    set max-fafs base-max-fafs
    set waiting 0
    set death-burgers base-burgers
    ; cree un agent 'sphere de perception'
    hatch-Perceptions 1 [
      set color c
      set my-range 2 * base-perception
      set agt myself
      ask myself [ set percept self ]
      set size 0
    ]
  ]
end

;
; cree 'n' burgers
;
to new-random-burgers [ n ]
  ; position choisie aleatoirement
  let x random-xcor
  let y random-ycor
  ; cree n burgers autour de la position choisie
  create-burgers n [ init-burger x y wild-burger-min-nrj wild-burger-max-nrj Burgers -1 ]
  ; cree des burgers à la même position relative pour les 2 autres bases
  create-burgers n [ init-burger x + 60 y wild-burger-min-nrj wild-burger-max-nrj Burgers -1 ]
;  create-burgers n [ init-burger x + 80 y wild-burger-min-nrj wild-burger-max-nrj Burgers -1 ]
end

;
; cree 'n' burgers à la position de l'agent
;
to new-burgers [ x y n min-nrj max-nrj bb wi ]
  ; cree n burgers autour de l'agent
  hatch-Burgers n [ init-burger x y min-nrj max-nrj bb wi ]
end

to init-burger [ x y min-nrj max-nrj bb wi ]
  ; positions etalees autour du point choisi
  setxy x y
  set burger-breed bb
  set who-id wi
  set label ""
  set heading random 360
  fd random 3
  set size 1
  ; energie entre 50 et 100
  set energy min-nrj + random (max-nrj + 1 - min-nrj)
;  show word word burger-breed " / " who-id
end

;
; Creation de nouveaux obstacles
;
to new-random-walls
  ; 180 murs crees aleatoirement
  create-Walls 60 [
    let x random-xcor
    let y random-ycor
    ;; crée 3 murs symétriques
    init-wall x y
    init-wall x + 40 y
    init-wall x + 80 y
  ]
end

;; seules les bases peuvent créer de nouveaux murs
to new-wall [ a ]
  if (([breed] of a = Bases) and create-ok?) [
    ; on verifie que l'agent a suffisamment d'energie
    if ([energy] of a > wall-cost) [
      ; creation de l'agent
```

```
      hatch-Walls 1 [
        ; meme couleur et orientation que l'agent parent
        set color [color] of a
        set size 1
        ; on decale le nouvel agent
        fd 1
        ; initialisation de l'energie, du rayon de perception, de la vitesse
        set energy wall-nrj
      ]
      ; decremente l'energie de l'agent createur
      ask a [
        set energy energy - wall-cost
        set create-ok? false
      ]
    ]
  ]
end

to init-wall [ x y ]
  setxy x y
  set energy wall-nrj
  set color gray
  set size 1
end

;
; Creation d'un nouvel explorer par l'agent 'a'
;
to new-Explorer [ a ]
  ; on verifie que l'agent a suffisamment d'energie
  if (([energy] of a > explorer-cost) and create-ok?) [
    ; orientation choisie aleatoirement
    set heading random 360
    ; creation de l'agent
    hatch-Explorers 1 [
      ; initialisation de sa taille
      set size 2
      ; meme couleur et orientation que l'agent parent
      set color [color] of a
      set friend [friend] of a
      set ennemy [ennemy] of a
      set heading [heading] of a
      ; on decale le nouvel agent
      fd 1
      ; initialisation de l'energie, du rayon de perception, de la vitesse
      set energy explorer-nrj
      set detection-range explorer-perception
      set speed explorer-speed
      set my-bases Bases with [ color = [color] of a ]
      set label ""
      set mem0 0
      set mem1 0
      set mem2 0
      set mem3 0
      set mem4 0
      set mem5 0
      set death-burgers explorer-burgers
      set carrying-food? 0
      ; cree un agent 'sphere de perception'
      hatch-Perceptions 1 [
        set color [color] of a
        set my-range 2 * explorer-perception
        set agt myself
        ask myself [ set percept self ]
        set size 0
      ]
      ifelse (color = green)
      [ initGreenExplorer ]
      [ initRedExplorer ]
    ]
    ; decremente l'energie de l'agent createur
    ask a [
```

```
      set energy energy - explorer-cost
      set create-ok? false
    ]
  ]
end

;
; Creation d'un nouveau rocket-launcher par l'agent 'a'
;
to new-RocketLauncher [ a ]
  ; on verifie que l'agent a suffisamment d'energie
  if (([energy] of a > rocket-launcher-cost) and create-ok?) [
    ; orientation choisie aleatoirement
    set heading random 360
    ; creation de l'agent
    hatch-RocketLaunchers 1 [
      ; initialisation de sa taille
      set size 2
      ; meme couleur et orientation que l'agent parent
      set color [color] of a
      set friend [friend] of a
      set ennemy [ennemy] of a
      set heading [heading] of a
      ; on decale le nouvel agent
      fd 1
      ; initialisation de l'energie, du rayon de perception, de la vitesse
      set energy rocket-launcher-nrj
      set detection-range rocket-launcher-perception
      set speed rocket-launcher-speed
      ; initialement 30 missiles, 100 au maximum
      set nb-missiles rocket-launcher-nb-missiles
      set max-missiles rocket-launcher-max-missiles
      ; initialement 30 missiles, 100 au maximum
      set nb-fafs rocket-launcher-nb-fafs
      set max-fafs rocket-launcher-max-fafs
      set waiting 0
      set my-bases Bases with [ color = [color] of a ]
      set label ""
      set mem0 0
      set mem1 0
      set mem2 0
      set mem3 0
      set mem4 0
      set mem5 0
      set death-burgers rocket-launcher-burgers
      set carrying-food? 0
      ; cree un agent 'sphere de perception'
      hatch-Perceptions 1 [
        set color [color] of a
        set my-range 2 * rocket-launcher-perception
        set agt myself
        ask myself [ set percept self ]
        set size 0
      ]
      ifelse (color = green)
      [ initGreenRocketLauncher ]
      [ initRedRocketLauncher ]
    ]
    ; decremente l'energie de l'agent createur
    ask a [
      set energy energy - rocket-launcher-cost
      set create-ok? false
    ]
  ]
end

;
; Creation d'un nouvel harvester par l'agent 'a'
;
to new-Harvester [ a ]
  ; on verifie que l'agent a suffisamment d'energie
  if (([energy] of a > harvester-cost) and create-ok?) [
```

```
    ; orientation choisie aleatoirement
    set heading random 360
    ; creation de l'agent
    hatch-Harvesters 1 [
      ; initialisation de sa taille
      set size 2
      ; meme couleur et orientation que l'agent parent
      set color [color] of a
      set friend [friend] of a
      set ennemy [ennemy] of a
      set heading [heading] of a
      ; on decale le nouvel agent
      fd 1
      ; initialisation de l'energie, du rayon de perception, de la vitesse
      set energy harvester-nrj
      set detection-range harvester-perception
      set speed harvester-speed
      set my-bases Bases with [ color = [color] of a ]
      set label ""
      set mem0 0
      set mem1 0
      set mem2 0
      set mem3 0
      set mem4 0
      set mem5 0
      set death-burgers harvester-burgers
      set carrying-food? 0
      set carried-walls-nrj []
      ; cree un agent 'sphere de perception'
      hatch-Perceptions 1 [
        set color [color] of a
        set my-range 2 * harvester-perception
        set agt myself
        ask myself [ set percept self ]
        set size 0
      ]
      ifelse (color = green)
      [ initGreenHarvester ]
      [ initRedHarvester ]
    ]
    ; decremente l'energie de l'agent createur
    ask a [
      set energy energy - harvester-cost
      set create-ok? false
    ]
  ]
end

;
; affiche ou non un label sur l'agent
;
to display-label
  set label ""
  ifelse (display? = "missiles")
  [ ifelse ((breed = RocketLaunchers) or (breed = Bases)) [ set label nb-missiles ][ set label "" ]]
  [ ifelse (display? = "fafs")
    [ ifelse ((breed = RocketLaunchers) or (breed = Bases)) [ set label nb-fafs ][ set label "" ]]
    [ ifelse (display? = "energy")
      [ set label round energy ]
      [ ifelse (display? = "carried-walls-nrj")
        [ ifelse (breed = Harvesters) [set label carried-walls-nrj][ set label ""]]
        [ ifelse (display? = "mem0")
          [ set label mem0 ]
          [ ifelse (display? = "mem1")
            [ set label mem1 ]
            [ ifelse (display? = "mem2")
              [ set label mem2 ]
              [ ifelse (display? = "mem3")
                [ set label mem3 ]
                [ ifelse (display? = "mem4")
                  [ set label mem4 ]
                  [ ifelse (display? = "mem5")
```

```
                    [ set label mem5 ]
                    [ if (display? = "carrying-food?")
                      [ ifelse ((breed = Bases) or (breed = RocketLaunchers))
                        [ set label "" ]
                        [ set label round carrying-food? ]
                      ]
                    ]
                  ]
                ]
              ]
            ]
          ]
        ]
      ]
    ]
  ]
end

;
; si l'agent n'a plus d'energie, il meurt
;
to mort
  if (energy <= 0) [
    ; crée des burgers en mourant
    new-burgers xcor ycor death-burgers wild-burger-min-nrj wild-burger-max-nrj breed who
    ; supprime le cercle de perception
    ask percept [die]
    ; meurt
    die
  ]
end

;
; prend le rocher 'b'
;
to take-wall [ b ]
  if ([breed] of b = Walls) [
    ;; seuls les agents de type "moissonneurs" peuvent soulever les rochers
    if ((breed = Harvesters) and (length carried-walls-nrj < 5)) [
      ; si on est suffisamment proche
      if (distance b <= 2) [
        set carried-walls-nrj fput [energy] of b carried-walls-nrj
        ; l'agent-rocher est tue
        ask b [ die ]
      ]
    ]
  ]
end

;
; depose le rocher devant l'agent
;
to drop-wall
  if (not empty? carried-walls-nrj) [
    hatch-Walls 1 [
      set heading [heading] of myself
      set size 1
      set color [color] of myself
      set energy first [carried-walls-nrj] of myself
      set carrying-food? 0
      set mem0 0
      set mem1 0
      set mem2 0
      set mem3 0
      set mem4 0
      set mem5 0
      set detection-range 0
      set speed 0
      set fd-ok? false
      set percept 0
      set my-bases 0
      set friend gray
```

```
      set ennemy gray
      fd 1
    ]
    set carried-walls-nrj remove-item 0 carried-walls-nrj
  ]
end

;
; prend la nourriture 'b'
;
to take-food [ b ]
  if ([breed] of b = Burgers) [
    ;; seuls les agents de type "moissonneurs" peuvent recolter la nourriture
    if (breed = Harvesters) [
      ; si on est suffisamment proche
      if (distance b <= 2) [
        without-interruption [
          ; on augmente la quantite de nourriture transportee
          set carrying-food? carrying-food? + [energy] of b
          ; l'agent-nourriture est tue
          ask b [ die ]
        ]
      ]
    ]
  ]
end

;
; donne la quantite 'c' de nourriture a l'agent 'agent'
;
to give-food [ agent c ]
  ; on ne peut donner de la nourriture qu'à un autre moissonneur ou à la base
  if ((agent != nobody) and (([breed] of agent = Explorers) or ([breed] of agent = Harvesters) or ([breed] of
agent = Bases))) [
    ; si les 2 agents sont suffisamment proches l'un de l'autre
    if (distance agent <= 2) and (c <= carrying-food?) [
      ; incremente la nourriture de l'autre agent
      ask agent [ set carrying-food? carrying-food? + c ]
      ; decremente sa propre nourriture
      set carrying-food? carrying-food? - c
      if (carrying-food? < 0) [show "pb d'échange de nourriture"]
    ]
  ]
end

;
; plante 'nb' graines dans le sol
; - on ne peut pas avoir plus de 'max-seeds' graines dans un patch
; - il faut consommer 'seed-cost' unites de 'carrying-food?' pour planter une graine
;
to plant-seeds [ c nb ]
  ; compte le nombre de graine ici
  let nb-seeds count Seeds-here
  ; evalue combien on peut planter de graines ici
  set nb min (list nb (max-seeds - nb-seeds) (carrying-food? / seed-cost))

  if (nb > 0) and (count Walls-here = 0) [
    ; plante les graines
    ask patch-here [
      sprout-Seeds nb [ set age 0 set shape "dot" set label "" set size 1 set friend c  ]
    ]
    ; c'est preleve sur la nourriture
    set carrying-food? carrying-food? - nb * seed-cost
    if (carrying-food? < 0) [show "pb de plantation"]
  ]
end

;
; assure la croissance des graines
; - au bout de 'maturation-time' ticks, produit un burger avec une energie comprise
;   entre 'seeded-burger-min-nrj' et 'seeded-burger-max-nrj'
;
```

```
to grow-seed
  set age age + 1
  set color scale-color friend age 0 maturation-time
  if (age = maturation-time) [
    new-burgers xcor ycor 1 seeded-burger-min-nrj seeded-burger-max-nrj breed who
    die
  ]
end

;
; donne la quantite 'nrj' d'energie a l'agent 'agent'
;
to give-energy [ agent nrj ]
  if (breed = Bases) [
    if (agent != nobody) [
      ; si les 2 agents sont suffisamment proches l'un de l'autre
      if (distance agent <= 2) and (nrj <= energy) [
        ; incremente l'energie de l'autre agent
        ask agent [ set energy energy + nrj ]
        ; decremente sa propre energie
        set energy energy - nrj
      ]
    ]
  ]
end

;
; procedure reservee aux bases pour convertir la nourriture
; que les agents ont ramenee en energie
;
to convert-food-into-energy
  ; incremente l'energie
  set energy energy + carrying-food?
  ; supprime la nourriture
  set carrying-food? 0
end

;
; renvoie un des agents présent devant dans un cone d'ouverture 135° et jusqu'à une distance d
; (hormis les burgers, les graines et les agents de perception)
to-report free-ahead? [ d ]
  set d max (list 1 d)
  report one-of other turtles in-cone d 135 with [ breed != Burgers and breed != Seeds and breed != Perceptions
and breed != Missiles and breed != Fafs ]
end

;
; avance vers l'avant
;
to forward-move [ d ]
  without-interruption [
    ; si l'agent ne s'est pas deja deplace au cours de ce cycle
    if (fd-ok?) [
      ; il inhibe son deplacement pendant le tour
      set fd-ok? false
      ; limite le déplacement à la vitesse de l'agent
      set d min (list d speed)
      ; est-ce qu'il y a un agent devant
      let a free-ahead? d
      ; si la voie est libre
      ifelse (a = nobody) [
        ; il peut avancer...
        fd d
        ; il peut écraser les plants de burgers (sauf si c'est un harvester)
        if (breed != Harvesters) [ ask Seeds-here with [age > 100][ set age age - 100 ]]
      ]
      ; sinon, il y a collision avec dégâts sur l'obstacle et sur le robot
      [
        ask a [ set energy energy - collision-damage ]
        ifelse ([breed] of a = Bases)
        [ set energy 0 ]
        [ set energy energy - collision-damage ]
```

```
      ]
    ]
  ]
end

;
; effectue un mouvement aleatoire
;
to random-move
  ; choisit une orientation aleatoire entre -45 et +45 degres
  rt random 91 - 45
  ; avance vers l'avant
  if (free-ahead? speed = nobody) [forward-move speed]
end

;
; lance une nouvelle rocket dans la direction 'dir'
;
to launch-rocket [ dir ]
  ; si l'agent a encore des missiles et qu'il respecte le delai d'attente
  if ((waiting = 0) and (nb-missiles > 0)) [
    ; on decremente le nombre de missiles de l'agent
    set nb-missiles nb-missiles - 1
    ; impose un delai d'attente de 5 entre 2 lancers de missiles
    ifelse (breed != bases)
    [ set waiting rocket-launcher-waiting ]
    [ set waiting base-waiting ]

    ; on cree un nouveau missile
    hatch-Missiles 1 [
      ; pas de label pour le missile
      set label ""
      ; taille 1
      set size 1
      ; portee 20
      set my-range missile-range
      ; en se dirigeant dans la direction 'dir'
      set heading dir
      ; a la vitesse de 1
      set speed missile-speed
      ; on decale le missile par rapport a l'agent
      fd 0.5
    ]
  ]
end

;
; procedure de "guidage" des missiles
;
to go-missile
  ; recupere la liste des agents a une distance inferieure a 1.5 dans un cone de 180 degres
  let t turtles in-cone 1.5 180 with [ (breed != Missiles) and (breed != Fafs) and (breed != Burgers) and
(breed != Perceptions) and (breed != Seeds) ]
  ; s'il y a des agents
  if any? t [
    ; on decremente leur energie de 100 si ce n'est pas une base, de 20 si c'est une base
    ask t [ ifelse (breed != Bases) [ set energy energy - missile-robot-damage ][ set energy energy - missile-
base-damage ] ]
    ; le missile est detruit
    die
  ]
  ; si la portee du missile devient nulle...
  ifelse my-range < 0
    ; ... alors il est detruit
    [ die ]
    ; sinon il avance a la vitesse speed, et sa portee est diminuee d'autant
    [  ; avance tout droit à la vitesse speed
      fd speed set my-range my-range - speed
    ]
end
```

```
;
; possibilite de creer 'n' nouveaux missiles pour un rocket-launcher ou pour la base
; cout = 10 pour le creer
; degats = 100 pour un robot / 20 pour une base
;
to new-missile [ n ]
  if ((breed = RocketLaunchers) or (breed = Bases)) [
    ; si on n'atteint pas le max de missiles autorises et qu'on a suffisamment d'energie
    if (nb-missiles + n <= max-missiles) and (energy > missile-cost * n) [
      ; le nb de missiles est incremente de 'n'
      set nb-missiles nb-missiles + n
      ; chaque missile coûte 10 points d'energie a fabriquer
      set energy energy - missile-cost * n
    ]
  ]
end

;
; lance une nouvelle rocket de type faf en direction de la cible t
;
to launch-faf [ t ]
  ; si l'agent a encore des fafs et qu'il respecte le delai d'attente
  if ((waiting = 0) and (nb-fafs > 0)) [
    ; on decremente le nombre de fafs de l'agent
    set nb-fafs nb-fafs - 1
    ; impose un delai d'attente de 5 entre 2 lancers de fafs
    ifelse (breed != bases)
    [ set waiting rocket-launcher-waiting ]
    [ set waiting base-waiting ]

    ; on cree un nouveau faf
    hatch-Fafs 1 [
      ; pas de label pour le faf
      set label ""
      ; taille 1
      set size 3
      set color yellow
      ; portee 20
      set my-range faf-range
      ; en se dirigeant dans la direction de la cible
      if (t != nobody) [
        set heading towards t
      ]
      ; a la vitesse de 1
      set speed faf-speed
      ; on decale le faf par rapport a l'agent
      fd 0.5
      ; vérouille la cible
      set target t
    ]
  ]
end

;
; procedure de "guidage" des missiles
;
to go-faf
  ; recupere la liste des agents a une distance inferieure a 1.5 dans un cone de 180 degres
  let t turtles in-cone 1.5 180 with [ (breed != Missiles) and (breed != Fafs) and (breed != Burgers) and
(breed != Perceptions) and (breed != Seeds) ]
  ; s'il y a des agents
  if any? t [
    ; on decremente leur energie de la quantité prédéfinie
    ask t [ ifelse (breed != Bases) [ set energy energy - faf-robot-damage ][ set energy energy - faf-base-
damage ] ]
    ; le faf est detruit
    die
  ]
  ; si la portee du missile devient nulle...
  ifelse my-range < 0
    ; ... alors il est detruit
    [ die ]
```

```
    ; sinon il avance a la vitesse speed, et sa portee est diminuee d'autant
    [
      let h heading
      if (target != nobody) [
        set heading towards target
      ]
      ; avance tout droit à la vitesse speed
      fd speed set my-range my-range - speed
    ]
end


;
; possibilite de creer 'n' nouveaux missiles pour un rocket-launcher ou pour la base
; cout = 10 pour le creer
; degats = 100 pour un robot / 20 pour une base
;
to new-faf [ n ]
  ; si on n'atteint pas le max de missiles autorises et qu'on a suffisamment d'energie
  if (nb-fafs + n <= max-fafs) and (energy > faf-cost * n) [
    ; le nb de missiles est incremente de 'n'
    set nb-fafs nb-fafs + n
    ; chaque missile coûte 10 points d'energie a fabriquer
    set energy energy - faf-cost * n
  ]
end

;
; renvoie les agents de type Burger dans le rayon de perception de l'agent
;
to-report perceive-food
  report Burgers in-radius detection-range
end

;
; renvoie les missiles de type faf dans le rayon de perception de l'agent
;
to-report perceive-fafs
  report Fafs in-radius detection-range
end

;
; renvoie les agents de type Burger dans le rayon de perception de l'agent
;
to-report perceive-food-in-cone [ a ]
  report Burgers in-cone detection-range a
end

;
; renvoie les agents de type Seed dans le rayon de perception de l'agent
; les graines ne sont perceptibles que quand elle ont un age > 500
;
to-report perceive-seeds [ c ]
  ifelse (c = friend)
  [ report Seeds in-radius detection-range with [ friend = c ]]
  [ report Seeds in-radius detection-range with [ (friend = c) and (age > 500) ]]
end

;
; renvoie les agents de type Seed dans le rayon de perception de l'agent
; les graines ne sont perceptibles que quand elle ont un age > 500
;
to-report perceive-seeds-in-cone [ a c ]
  ifelse (c = friend)
  [ report Seeds in-cone detection-range a with [ friend = c ]]
  [ report Seeds in-cone detection-range a with [ (friend = c) and (age > 500) ]]
end

;
; renvoie les agents de type Wall dans le rayon de perception de l'agent
;
to-report perceive-walls
```

```
  report Walls in-radius detection-range
end

;
; renvoie les agents de type Wall dans le rayon de perception de l'agent
;
to-report perceive-walls-in-cone [ a ]
  report Walls in-cone detection-range a
end

;
; renvoie les agents de couleur 'c' et de type 'Explorer' ou 'RocketLauncher' dans le rayon de perception de
l'agent
;
to-report perceive-robots [ c ]
  report turtles in-radius detection-range with [ (color = c) and ((breed = Explorers) or (breed =
RocketLaunchers) or (breed = Harvesters))]
end

;
; renvoie les agents de couleur 'c' et de type 'Explorer' ou 'RocketLauncher' ou 'Harvesters' dans le rayon de
perception de l'agent
;
to-report perceive-robots-in-cone [ c a ]
  report turtles in-cone detection-range a with [ (color = c) and ((breed = Explorers) or (breed =
RocketLaunchers) or (breed = Harvesters))]
end

;
; renvoie les agents de couleur 'c' et de type b dans le rayon de perception de l'agent
;
to-report perceive-specific-robots [ c b  ]
  report turtles in-radius detection-range with [ (color = c) and (breed = b)]
end

;
; renvoie les agents de couleur 'c' et de type b dans le rayon de perception de l'agent
;
to-report perceive-specific-robots-in-cone [ c b a ]
  report turtles in-cone detection-range a with [ (color = c) and (breed = b)]
end

;
; renvoie les bases de couleur 'c' dans le rayon de perception de l'agent
;
to-report perceive-base [ c ]
  report turtles in-radius detection-range with [ (color = c) and (breed = Bases) ]
end

;
; renvoie les bases de couleur 'c' dans le rayon de perception de l'agent
;
to-report perceive-base-in-cone [ c a ]
  report turtles in-cone detection-range a with [ (color = c) and (breed = Bases) ]
end

;
; renvoie les agents de couleur 'c1' ou 'c2' et de type 'Explorer' ou 'RocketLauncher' dans le rayon de
perception de l'agent
;
to-report perceive-robots2 [ c1 c2 ]
  report turtles in-radius detection-range with [ ((color = c1) or (color = c2)) and ((breed = Explorers) or
(breed = RocketLaunchers) or (breed = Harvesters))]
end

;
; renvoie les agents de couleur 'c1' ou 'c2' et de type 'Explorer' ou 'RocketLauncher' ou 'Harvesters' dans le
rayon de perception de l'agent
;
to-report perceive-robots2-in-cone [ c1 c2 a ]
  report turtles in-cone detection-range a with [ ((color = c1) or (color = c2)) and ((breed = Explorers) or
(breed = RocketLaunchers) or (breed = Harvesters))]
```

```
end

;
; renvoie les agents de couleur 'c1' ou 'c2' et de type b dans le rayon de perception de l'agent
;
to-report perceive-specific-robots2 [ c1 c2 b  ]
  report turtles in-radius detection-range with [ ((color = c1) or (color = c2)) and (breed = b)]
end

;
; renvoie les agents de couleur 'c1' ou 'c2' et de type b dans le rayon de perception de l'agent
;
to-report perceive-specific-robots2-in-cone [ c1 c2 b a ]
  report turtles in-cone detection-range a with [ ((color = c1) or (color = c2)) and (breed = b)]
end

;
; renvoie les bases de couleur 'c1' ou 'c2' dans le rayon de perception de l'agent
;
to-report perceive-base2 [ c1 c2 ]
  report turtles in-radius detection-range with [ ((color = c1) or (color = c2)) and (breed = Bases) ]
end

;
; renvoie les bases de couleur 'c1' ou 'c2' dans le rayon de perception de l'agent
;
to-report perceive-base2-in-cone [ c1 c2 a ]
  report turtles in-cone detection-range a with [ ((color = c1) or (color = c2)) and (breed = Bases) ]
end

;
; realise l'affichage d'un cercle autour de l'agent pour visualiser son rayon de perception
;
to go-perception
  ; si on demande l'affichage du rayon de perception
  ifelse (display-range?) [
    ; change la taille de l'agent
    set size my-range
    ; les coordonnees suivent celles de l'agent auquel l'agent de perception appartient
    ifelse ( agt != nobody ) [
      set xcor [xcor] of agt
      set ycor [ycor] of agt
    ]
    [ die ]
  ]
  ; ... sinon, taille 0 pour ne pas voir le cercle
  [ set size 0 ]
end
```