```
globals [
  percent-happy   ;; variables ilustrées dans nos graphes en pourcantage
  percent-density
]

turtles-own [
  happy?      ;; Pour chaque turtle, on a la  condition happy?
  yellow-nearby  ;; Couleur proche de la turtle en fonction d'un certain rayon
  white-nearby
  red-nearby
  similar-nearby ;; couleur similaire
  number-neighbors ;; Turtle dans les 8 cases autour
]

to setup
  clear-all

  ;; création aléatoir de turtles rouges sur les patches
  ask n-of number-red-resident patches
    [ sprout 1
      [ set color red ] ]
  ;; création aléatoir de turtles yellow sur les patches
   ask n-of number-yellow-activity patches
  [sprout 1
  [set color yellow] ]
;; création aléatoir de turtles white sur les patches
  ask n-of number-white-industry patches
  [sprout 1
  [set color white] ]
  ;; affectation de la forme carée
  ask turtles [ set shape "square"]
  update-variables
  reset-ticks
end


to go
  if percent-happy = 100  [ stop ] ;; le programme s'arrète lorsque on atteint un niveau de satisfaction
  move-unhappy-turtles  ;; Différentes procédures appelées pendant le fonctionnement d'un tick
  update-variables
  density
  tick
end


to move-unhappy-turtles ;; lorsques les turtles ne sont pas happy, elles bougent aléatoirement sur un autre
patch
  ask turtles with [happy? = FALSE]
    [ find-new-spot ]
end

to find-new-spot ;; fonction pour trouver un nouveau patch aléatoirement avec leurs degrés de libertés
  rt random-float 360 ;; permet une rotation d'un angle compris entre 0 et 360 degré
  fd random-float 30  ;; permet d'avancer de 30 patch
  if any? other turtles-here
    [ find-new-spot ]  ;; seulement si le patch n'est pas déja occupé
  move-to patch-here
end

to update-variables ;; permet de garder les couleurs attribuées préalablement
  update-turtles
  update-globals
end

to update-turtles

  ask turtles with [color = red] [  ;;  par rapport aux résidents ( rouge )
    ;; dans les lignes qui suivent, on compte le nombre des turtles dans un certain rayon ( qui peut varier)
    set similar-nearby count (turtles in-radius radius-resident)
      with [color = [color] of myself]
      set yellow-nearby count (turtles in-radius radius-resident)
      with [color = yellow]
      set white-nearby count (turtles in-radius radius-resident)
      with [color = white]

    set happy? yellow-nearby >= (resident-wants-activity )  and white-nearby <= (resident-accepts-industry)
    ;; condition pour que les résidents soient happy ( une attraction avec les activités mais une répulsion des
industries)
  ]
     ask turtles with [color = white] [ ;; par rapport aux industries ( blanc )
        ;; dans les lignes qui suivent, on compte le nombre des turtles dans un certain rayon ( qui peut
varier)
       set similar-nearby count (turtles in-radius radius-industry)
      with [color = [color] of myself]
      set red-nearby count (turtles in-radius radius-industry)
```

```
      with [color = red]

    set happy? similar-nearby >= (industry-wants-industry) and industry-accepts-resident >= ( red-nearby )
     ;; condition pour que les industries soient happy ( une attraction avec les industries mais une répulsion
des résidents)
  ]
       ask turtles with [color = yellow] [  ;; par rapport aux activités ( jaune)
         ;; dans les lignes qui suivent, on compte le nombre des turtles dans un certain rayon ( qui peut
varier)
         set similar-nearby count (turtles in-radius radius-activity)
      with [color = [color] of myself]
      set white-nearby count (turtles in-radius radius-activity)
      with [color = white]
      set red-nearby count (turtles in-radius radius-activity)
      with [color = red]

    set happy? white-nearby >= ( activity-needs-industry ) and red-nearby >= ( activity-wants-resident )
    ;; condition pour que les activités soient happy ( une attraction avec les industries et une attraction
avec les résidents)
  ]

end

to density
  ;; Méthode pour calculer la densité de toutes les turtles en pourcentage
  let tot 0
  ask turtles [
  set number-neighbors count turtles-on neighbors / 8 ]
  set tot sum   [number-neighbors] of turtles
  let moy-density tot / (count turtles)
  set percent-density moy-density * 100
end

to update-globals
  ;; méthodes pour calculer le taux de satisfaction des turtles
  set percent-happy (count turtles with [happy? = TRUE]) / (count turtles) * 100
end
```