Projet : « Circuler dans les rues de MANHATTAN »

Maker Khaoula

L3 MASS parcours ESD

Année scolaire : 2020-2021

INTRODUCTION :

Le projet traite la circulation dans les rues de Manhattan, et donc le but sera de simuler les déplacements de véhicules dans des rues qui sont perpendiculaires

Pour réaliser ce projet on va devoir traiter la simulation de mobilité pour chacune des étapes suivantes :

- Simuler le réseau des rues à l'aide de patches NetLogo.

- Créer et positionner des véhicules dans les rues de Manhattan.

- Simuler le déplacement aléatoire des véhicules dans les rues de Manhattan.

- Garantir le fait que chaque véhicule possède un carrefour de départ mais aussi un carrefour d'arrivée.

- Optimiser le déplacement des véhicules dans les rues de Manhattan.

Etape 1 :

Dans cette première étape il s'agit de simuler le réseau des rues à l'aide de patches NETLOGO , du coup on va commencer par définir les trois variables globales qui représentent respectivement la **taille** d'un bloc d'habitation , l'ensemble des patches correspondant aux **rues** et l'ensemble des patches correspondant aux **carrefours :**

```
globals [
  taille-bloc
  les-rues
  les-carrefours
]
```

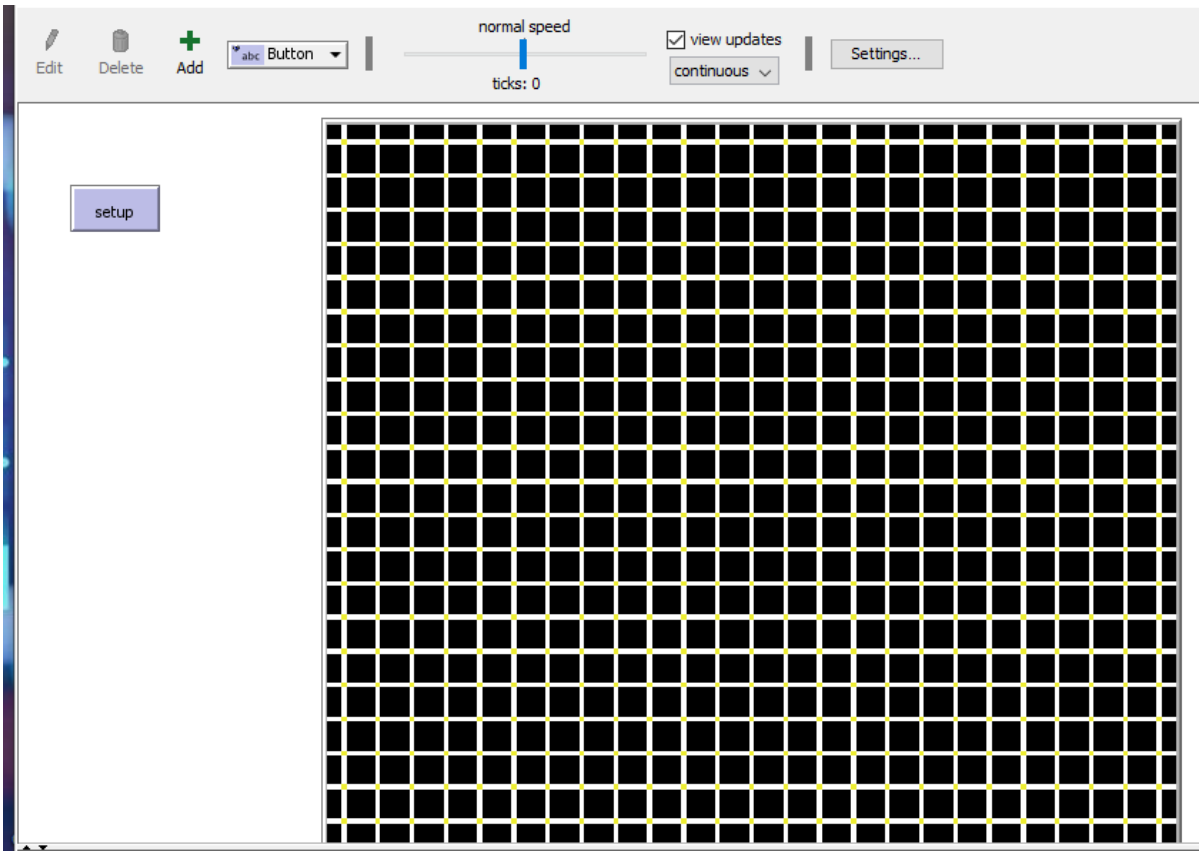
Ensuite on va compléter la procédure Setup qui permet de visualiser les rues en blancs , les bloc d'habitation en noire et les carrefours en jaune .

```
to setup
  clear-all
  set taille-bloc 7
  ask patches [set pcolor black] ; initialement
tous les patches sont noirs
  set les-rues patches with [pycor mod taille-bloc
= 0 or pxcor mod taille-bloc = 0 ]
  ask les-rues [set pcolor white] ;; afficher les
rues en blanc
  set les-carrefours patches with [pxcor mod
taille-bloc = 0 and pycor mod taille-bloc = 0 ]
  ask les-carrefours [set pcolor yellow] ;;
afficher les carrefours en jaune
  reset-ticks
end
```

On a supposé ici que la taille d'un bloc est 7 .

Pour exécuter ce code, on a créé un **bouton setup** dans l'interface Netlogo , et on a configuré le setting de l'interface graphique avec **origine center** , **max-pxcor** = 38 ou 87 et **max-pycor** = 38 ou 87 .

Voici le résultat de l'exécution du code :



Etape 2 :

Dans cette étape on va créer et positionner des véhicules dans les rues de **Manhattan**.

Premièrement, on va représenter les véhicules par une **turtle Netlgo** et les positionner sur un carrefour de départ qui est choisi au hasard à l'aide de la variable propre **carrefour-depart** , en positionnant au plus un véhicule par carrefour . Ensuite on va déterminer la proportion de carrefours initialement occupés par un des véhicules à l'aide d'une variable globale **densite-vehicules** avec un nombre de véhicules qui est constant . Le heading d'un véhicule prend aléatoirement les valeurs : `0°,90°,180°` ou `270°` , et lors des déplacements, le heading d'un véhicule ne peut changer que lorsque le véhicule est sur un carrefour .

voici le code qui va permettre d'exécuter toutes les opérations de création et de positionnement des véhicules :

```
globals [
  taille-bloc
  les-rues
  les-carrefours
]
```

```
breed [ vehicules vehicule ]

vehicules-own [ carrefour-depart carrefour-arrivee
temps-parcours]

to setup
  clear-all
  set taille-bloc 7
  ask patches [set pcolor black] ; initialement
tous les patches sont noirs
  set les-rues patches with [pycor mod taille-bloc
= 0 or pxcor mod taille-bloc = 0 ]
  ask les-rues [set pcolor white] ;; afficher les
rues en blanc
  set les-carrefours patches with [pxcor mod
taille-bloc = 0 and pycor mod taille-bloc = 0 ]
  ask les-carrefours [set pcolor yellow] ;;
afficher les carrefours en jaune
  reset-ticks
end

to setup-vehicules
   let nb-vehicules round (densite-vehicules * count
les-carrefours)
  ask n-of nb-vehicules les-carrefours [
       sprout-vehicules 1 [ ;; créer un véhicule sur
le patch-carrefour
       set carrefour-depart patch-here
           set shape "car"
           set size 6
           pen-up
       ]
   ]
   ask vehicule 0 [
       set color red
       set shape "truck"
       pen-down
   ]
end
```

Parallèlement, on va rajouter un slider qui représente **la densité vehicules** dans l'interface Netlogo .

Etape 3 :

Dans cette étape on va essayer de simuler le déplacement aléatoire des véhicules dans les rues de Manhattan .

Le déplacement du véhicule serait en ligne droite jusqu'à atteindre le prochain carrefour avec une vitesse constante , une fois le véhicule atteint un carrefour ,il peut soit continuer tout droit , faire demi-tour , ou tourner à droite ou à gauche .

Le code suivant résume la mobilité aléatoire :

```
to go
   ask vehicules [move-random]
   tick
end
to move-random ;; procédure exécutée par chaque véhicule
     if (pcolor = yellow ) [ ;; véhicule sur un carrefour
        face one-of neighbors4
     ]
     forward 1
end
```

Etape 4 :

Dans cette question, chaque véhicule possède un carrefour de départ mais aussi un carrefour d'arrivée .

Premièrement on va déclarer la variable propre qui va représenter le **carrefour-arrivee** aux véhicules , et on va l'initialiser aléatoirement dans la procédure **setup-vehicules .**

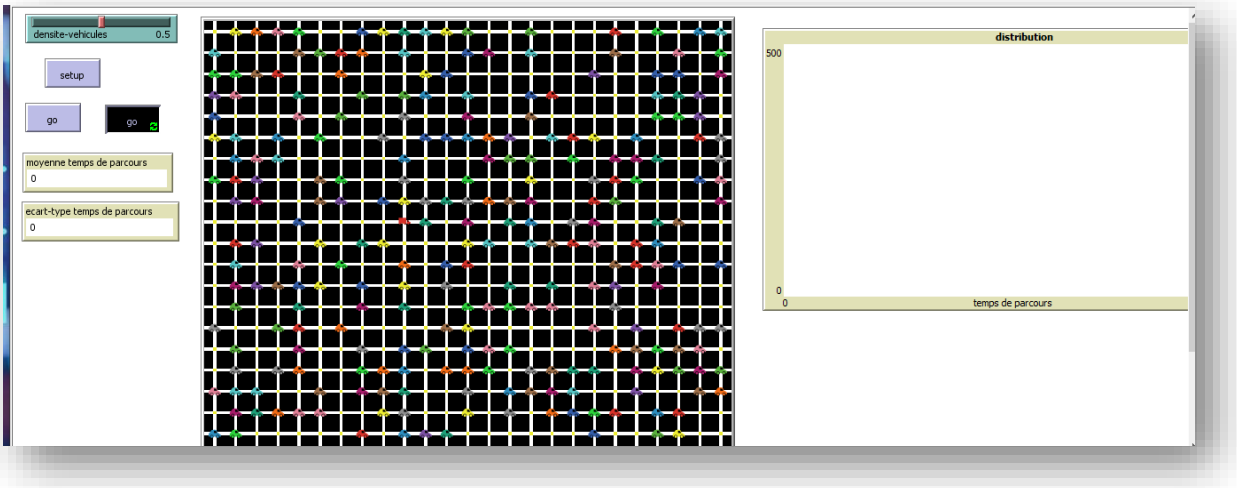
```
to go
  ask vehicules [ move-random-avec-arrivee ]
  tick
end
```

```
to move-random-avec-arrivee ;; procédure exécutée
par chaque véhicule
   if (patch-here  != carrefour-arrivee)
       [  ;; véhicule pas encore arrivé
          move-random
       ]
end
```

On crée une variable propre **temps-parcours** qui sert à connaitre le temps mis par chaque véhicule pour effectuer son trajet de son carrefour de départ à son carrefour d'arrivé , après on l'initialise dans la procédure **setup-vehicules** .Ensuite , on déclare deux composants de type **monitor** dans l'interface Netlogo , l'un va nous **indiquer la moyenne du temps de parcours** et l'autre va nous **indiquer l'écart-type** , et **un plot** qui représente **la distribution** du temps de parcours pour l'ensemble des véhicules .

Remarque :

On constate qu'après l'exécution du code que les voitures ne bougent pas , et que les deux boutons **go** and **go-forever** ne fonctionnent pas .



Etape 5 :

Dans cette étape on va optimiser le déplacement des véhicules dans les rues de Manhattan .

Une fois le véhicule atteint un carrefour ,il peut soit continuer tout droit , tourner à gauche ou à droite soit faire un demi-tour de façon à optimiser la distance le véhicule et son carrefour d'arrivée .

Dans le code suivant on va définir une nouvelle procédure **move-optimise** qui va prendre en compte la nouvelle façon pour se déplacer et des instructions comme **neighbors 4 , distance , min-one-of et patch at .**

```
to go
  ask vehicules [ move-optimise ]
  tick
end
to move-optimise ;; procédure exécutée par chaque
véhicule
   if (patch-here  != carrefour-arrivee)
      [ ;; vehicule pas encore arrivé
          if (pcolor = yellow ) [ ;; véhicule sur
un carrefour
              let but carrefour-arrivee
              face min-one-of neighbors4 [distance
but]
          ]
          forward 1
      ]
end
```

Maintenant on va réécrire le code complet qui est le suivant :

```
globals [
  taille-bloc
  les-rues
  les-carrefours
]
breed [ vehicules vehicule ]
vehicules-own [ carrefour-depart carrefour-arrivee temps-parcours]
to setup
  clear-all
  set taille-bloc 7
  ask patches [set pcolor black] ; initialement tous les patches sont noirs
```

```
  set les-rues patches with [ pycor mod taille-bloc = 0 or pxcor mod taille-bloc = 0 ]
  ask les-rues [set pcolor white] ;; afficher les rues en blanc
  set les-carrefours patches with [ pxcor mod taille-bloc = 0 and pycor mod taille-bloc = 0 ]
  ask les-carrefours [set pcolor yellow] ;; afficher les carrefours en jaune
  setup-vehicules
  reset-ticks
end



to setup-vehicules
   let nb-vehicules round (densite-vehicules * count les-carrefours)
  ask n-of nb-vehicules les-carrefours [
       sprout-vehicules 1 [ ;; créer un véhicule sur le patch-carrefour
       set carrefour-depart patch-here
       set carrefour-arrivee one-of les-carrefours
          set shape "car"
          set size 4
          pen-up
       ]
   ]
    ask vehicule 0 [
       set color red
       set shape "truck"
       pen-down
   ]
end

to go
  ;; ask vehicules [move-random]
  ;;ask vehicules [ move-random-avec-arrivee ]
```

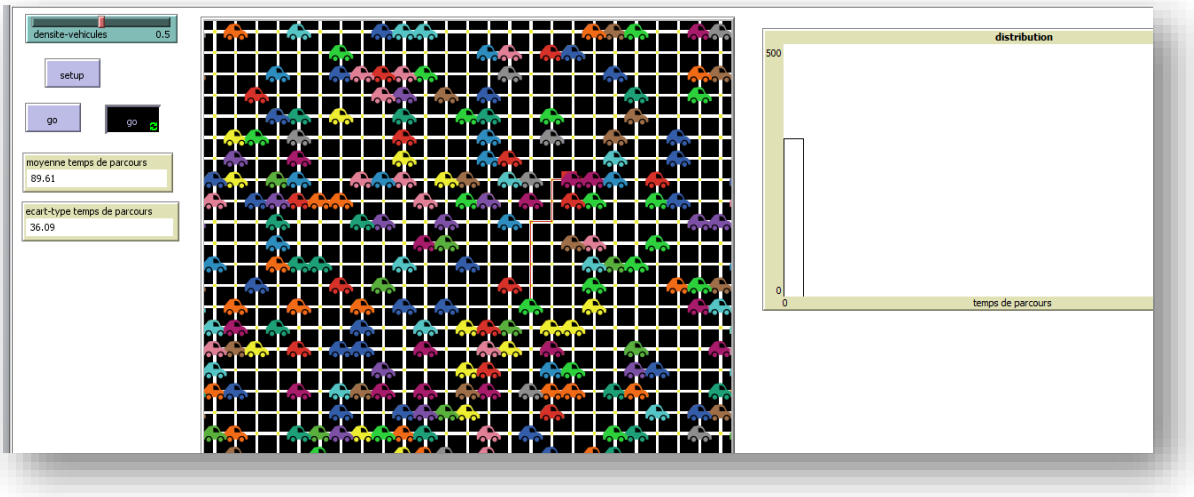
```
  ask vehicules [ move-optimise ]
  tick
end

to move-random ;; procédure exécutée par chaque véhicule
     if ( pcolor = yellow ) [ ;; véhicule sur un carrefour
        face one-of neighbors4
     ]
     forward 1
     set temps-parcours temps-parcours + 1
end

to move-random-avec-arrivee ;; procédure exécutée par chaque véhicule
   ifelse ( patch-here  = carrefour-arrivee)
       [ set size 8 ]
       [  ;; véhicule pas encore arrivé
          move-random
       ]
end

to move-optimise ;; procédure exécutée par chaque véhicule
   ifelse ( patch-here = carrefour-arrivee)
      [set size 8]
      [ ;; vehicule pas encore arrivé
          if ( pcolor = yellow ) [ ;; véhicule sur un carrefour
              let but carrefour-arrivee
      face min-one-of neighbors4 [distance but]
          ]
          forward 1
          set temps-parcours temps-parcours + 1
```

]

end

Remarque :

On constate maintenant que les boutons **go** et **go-forever** fonctionnent , et que les véhicules se déplacent aisément . De plus la distibution commence à se déssiner à partir de Ticks > 50000 .



Etape 6 :

Le but de mon extension est de simuler les feux de circulation qui changent de couleur entre rouge et vert .

J'ai commencé par dessiner dans les blocs des maisons et des personnes , puis des patches représentants les feux de circulation.

En plus, les véhicules changent de taille dès qu'ils traversent leurs carrefours d'arrivée.

Ces deux photos montrent le changement de tailles après plusieurs ticks :

Photo 1 :

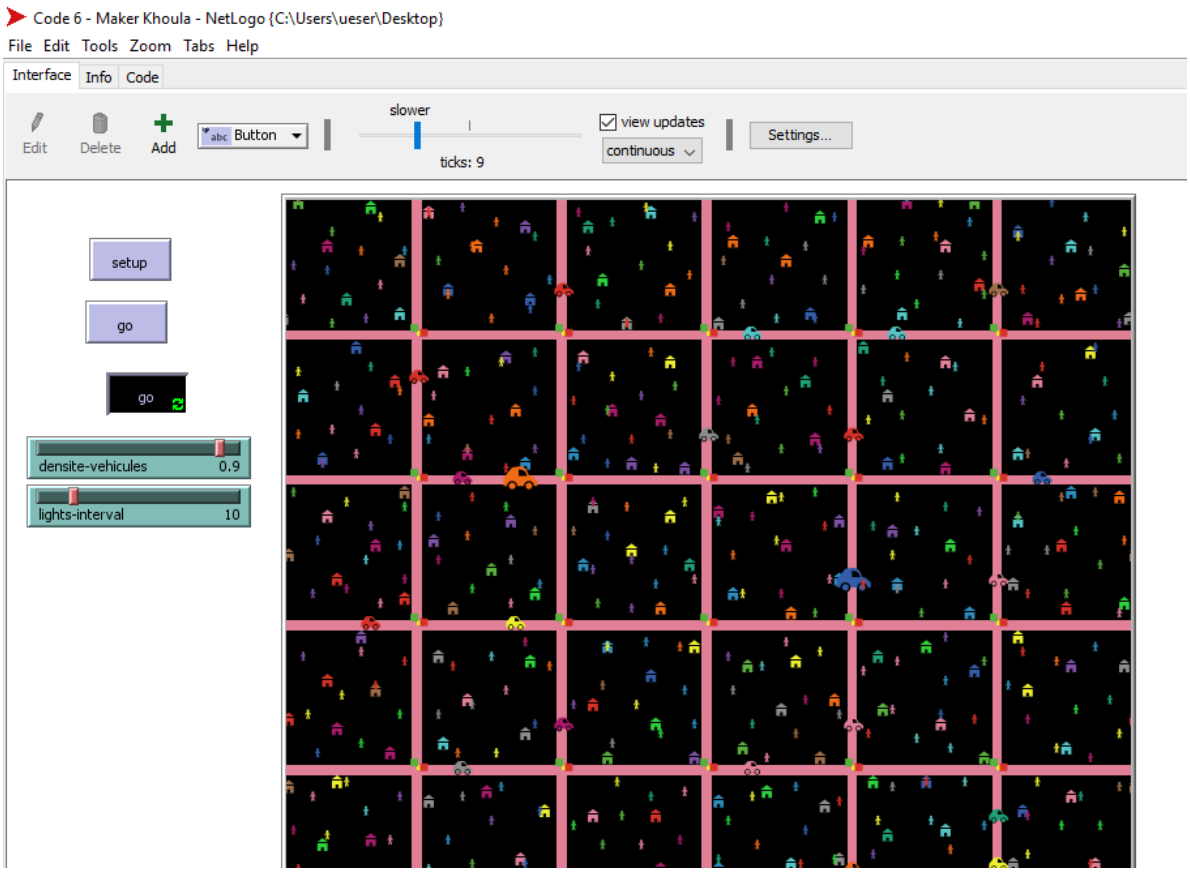
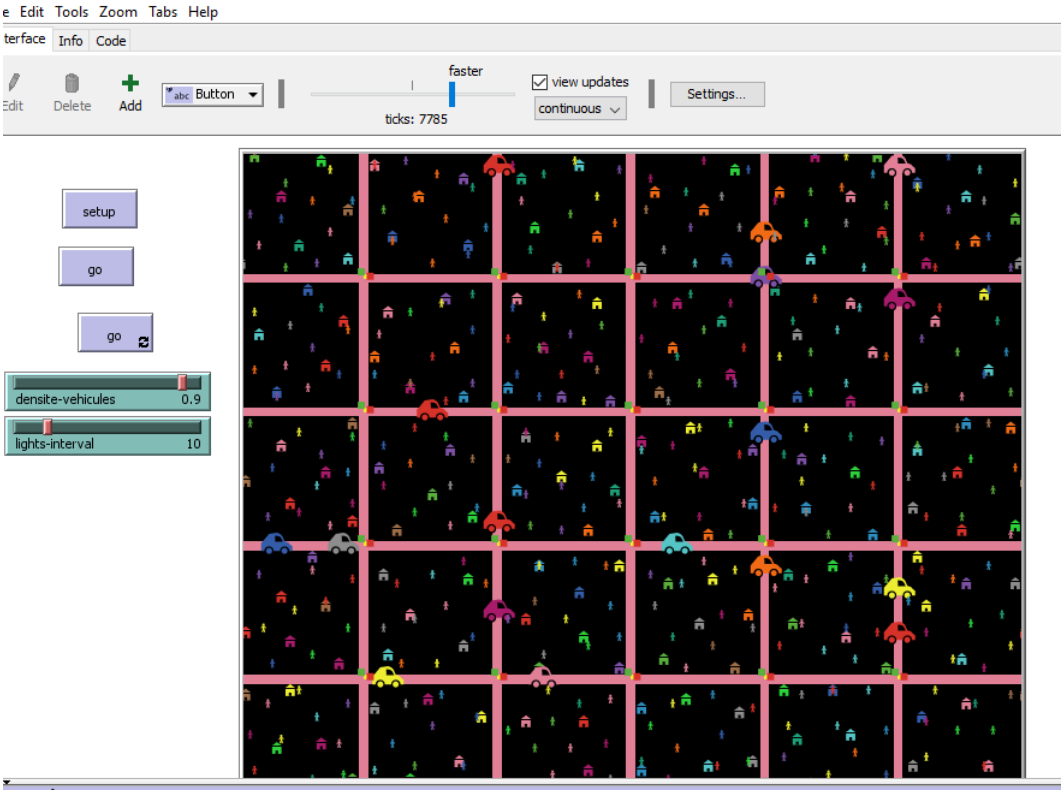

Code 6 - Maker Khoula - NetLogo {C:\Users\ueser\Desktop}
File Edit Tools Zoom Tabs Help
Interface Info Code
Edit
Delete
Add
Button
slower
ticks: 9
view updates
continuous
Settings...
setup
go
go
densite-vehicules 0.9
lights-interval 10

Photo 2 :



Voici le code 6 complet :

```
globals [
  taille-bloc
  les-rues
  les-carrefours
  redV
  greenV
  redH
  greenH
]

breed [ vehicules vehicule ]
```

```
breed[houses house]



breed[lightsL lightL]

breed[lightsD lightD]
breed[persons person]

vehicules-own [
  carrefour-depart
  carrefour-arrivee
  temps-parcours

]



to setup
  clear-all
  set taille-bloc 30

  ask patches [set pcolor black] ; initialement tous les patches sont noirs
  set les-rues patches with [ pycor mod taille-bloc = 0 or pxcor mod taille-bloc = 0 or pycor mod taille-bloc = 29 or pxcor mod taille-bloc = 29  ]
  ask les-rues [set pcolor pink] ;; afficher les rues en blanc
  set les-carrefours patches with [ pxcor mod taille-bloc = 0 and pycor mod taille-bloc = 0 ]
```

```
  ask les-carrefours [set pcolor yellow] ;; afficher les carrefours en jaune
  draw-houses
  place-lights
  place-people
  setup-vehicules
  reset-ticks
end



to setup-vehicules
   let nb-vehicules round (densite-vehicules * count les-carrefours)
  ask n-of nb-vehicules les-carrefours [
       sprout-vehicules 1 [ ;; créer un véhicule sur le patch-carrefour
       set carrefour-depart patch-here
       set carrefour-arrivee one-of les-carrefours
           set shape "car"
           set size 4
           pen-up
       ]
   ]



end

to go
  ;; ask vehicules [move-random]
  ;;ask vehicules [ move-random-avec-arrivee ]
```

```
  ask vehicules [ move-random ]
  control-traffic-lights
  tick
end




to control-traffic-lights
  if ticks mod (50 * lights-interval * greenH + 65 * lights-interval * redH ) = 0 [change-color lightsL "H" change-color lightsD "H"]

end



to change-color [lights D]

  ask one-of lights [
    ifelse color = red [
      ifelse D = "H" [
        set greenH greenH + 1
        ][
        set greenV greenV + 1]
        ]
    [
```

```
      ifelse D = "H" [
        set redH redH + 1][
        set redV redV + 1]
        ]

  ]

  ask lights [
    ifelse color = red [set color green][set color red]
  ]
end



to place-lights
 ask patches with [pxcor mod taille-bloc = 29 and pycor mod taille-bloc = 1 ] [
    sprout-lightsL 1 [
      set color red
      set shape "square"
      set size 2
    ]
  ]


  ask patches with [pxcor mod taille-bloc = 1 and pycor mod taille-bloc = 0] [
    sprout-lightsD 1 [
      set color green
      set shape "square"
```

```
      set size 2
    ]
  ]

  set greenH 0
  set redH 1
  set redV 0
  set greenV 1

end



to place-people
  ask patches with [pcolor = black] [
    if count neighbors with [pcolor = black] = 8 and not any? turtles in-radius 7 [
      sprout-persons 1 [

        set size 2

        set shape  "person"
        ]
      ]
    ]

end
```

```
to draw-houses
  ask patches with [pcolor = black] [
    if count neighbors with [pcolor = black] = 8 and not any? turtles in-radius 10 [
       sprout-houses 1    [ ;; créer un véhicule sur le patch-carrefour
        set shape "house"
        set size 3
        stamp
      ]
    ]
  ]

  ask houses [die]



end



to move-random ;; procédure exécutée par chaque véhicule
     if ( patch-here = carrefour-arrivee    ) [
       set size 7
     ]
     if ( pcolor = yellow ) [ ;; véhicule sur un carrefour

        face one-of neighbors4
     ]
     forward 1
```

```
    set temps-parcours temps-parcours + 1
end
```