Etude de la propagation des feux de forêt et percolation

Romain PIERRE | TIPE : Santé/Prévention 2022 | Numéro Concours : 42692

Sommaire

Introduction

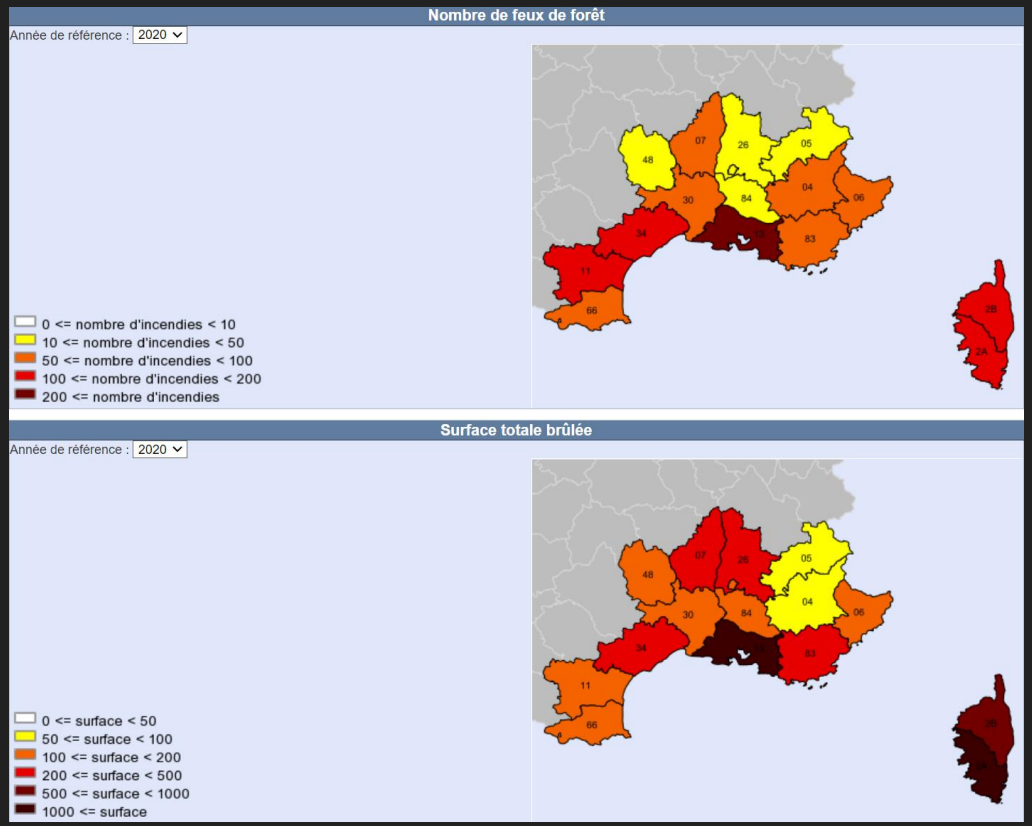
Feux de forêt

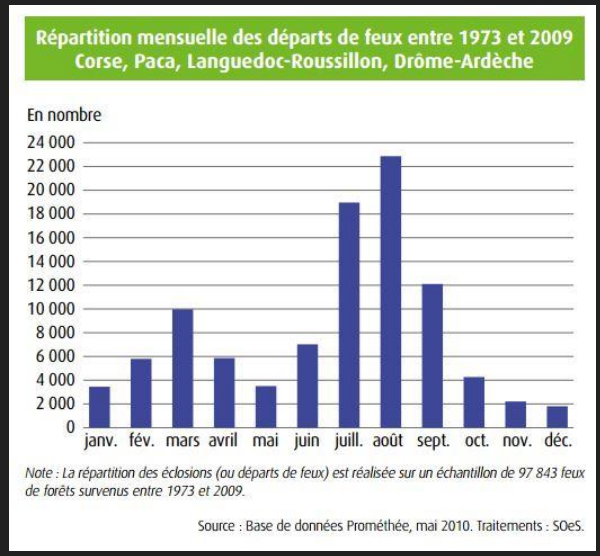
Les feux de forêt | Statistiques

4000 feux de forêt par an

6 fois la taille de la France qui brûle chaque année

Les feux de forêt | Origines





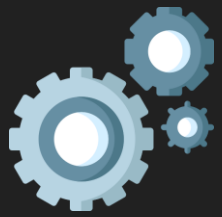
L'**inflammabilité** varie selon les régions et les saisons

⇒ Température/Sécheresse

⇒ Essence des arbres

Source : Base de donnée « Prométhée » (région méditerranéenne)

Les feux de forêt | Origines



Probabilité de transmission du feu
d'un arbre enflammé à son voisin

Les feux de forêt | Origines

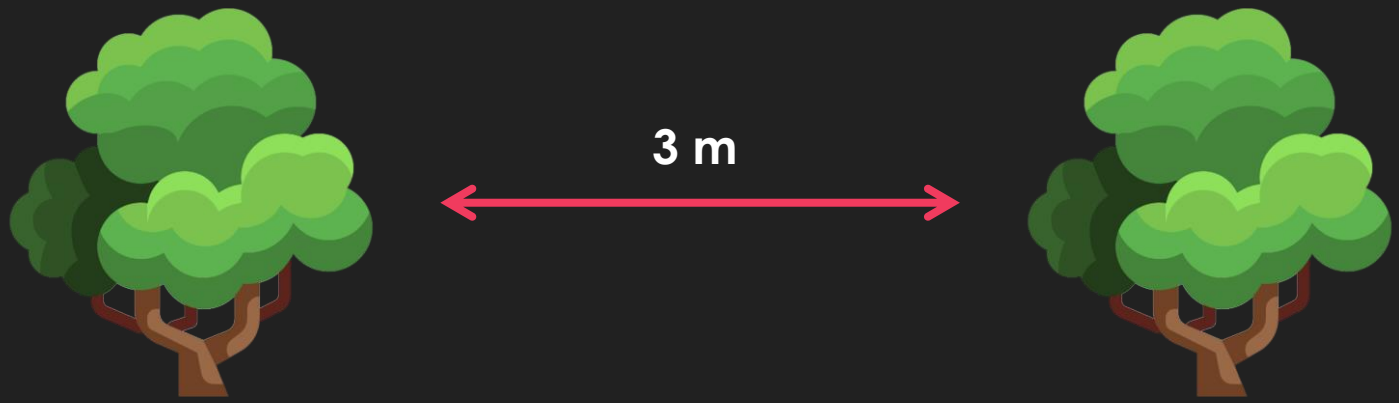
Code forestier :

On entend par **débroussaillement** les opérations de réduction des combustibles végétaux de toute nature dans le but de diminuer l'intensité et de **limiter la propagation des incendies**.

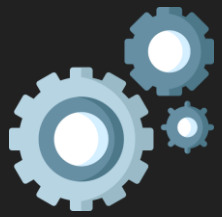
Ces opérations assurent une rupture suffisante de la continuité du couvert végétal.

[Ordonnance n°2012-92 du 26 janvier 2012 art. (V) Article L131-10]

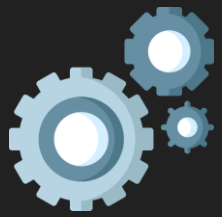
Les feux de forêt | Origines



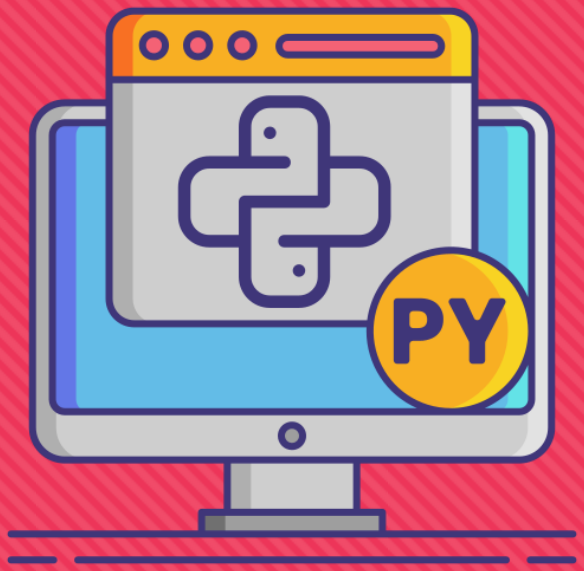
Les feux de forêt | Origines



Probabilité de transmission du feu
d'un arbre enflammé à son voisin



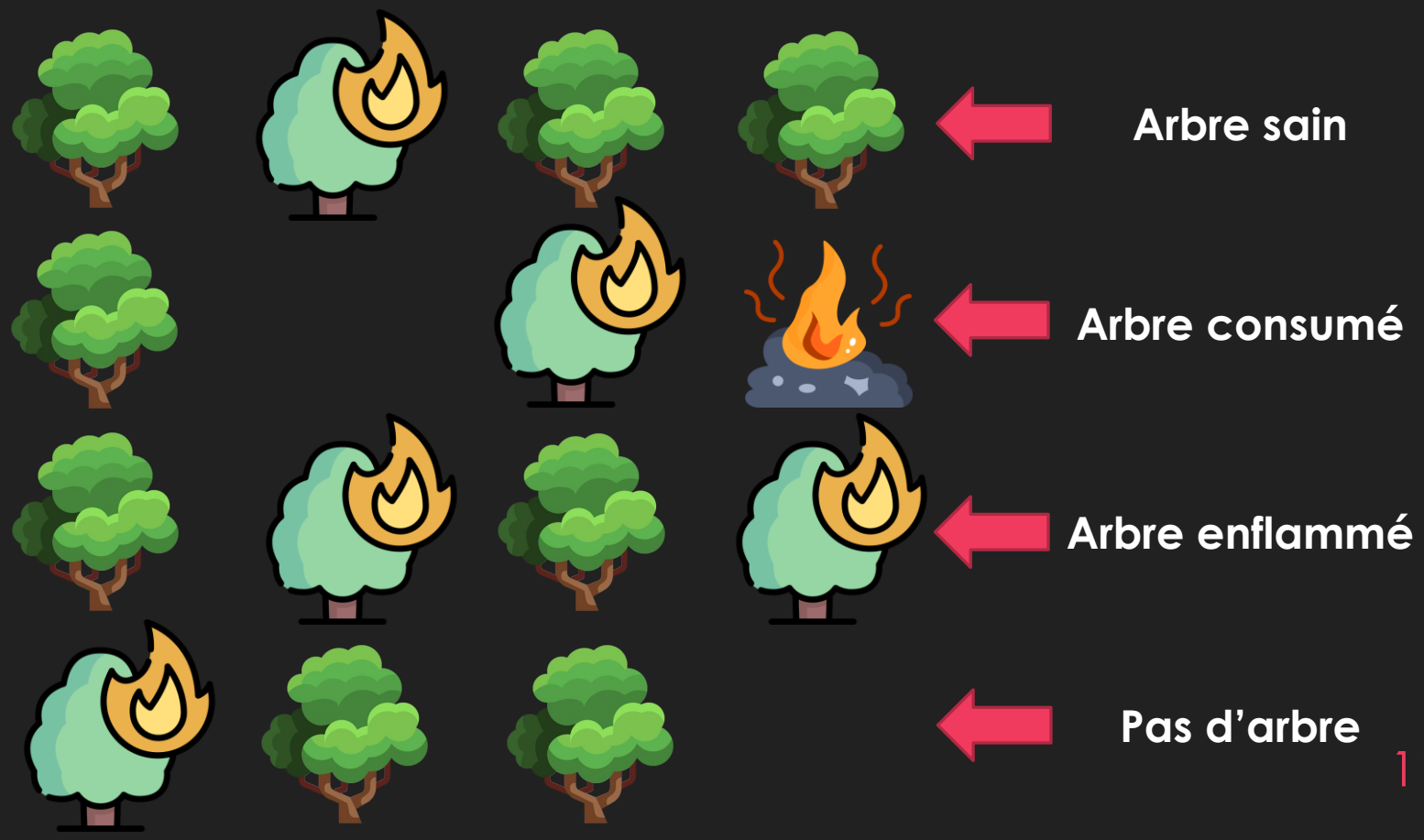
Densité de la forêt



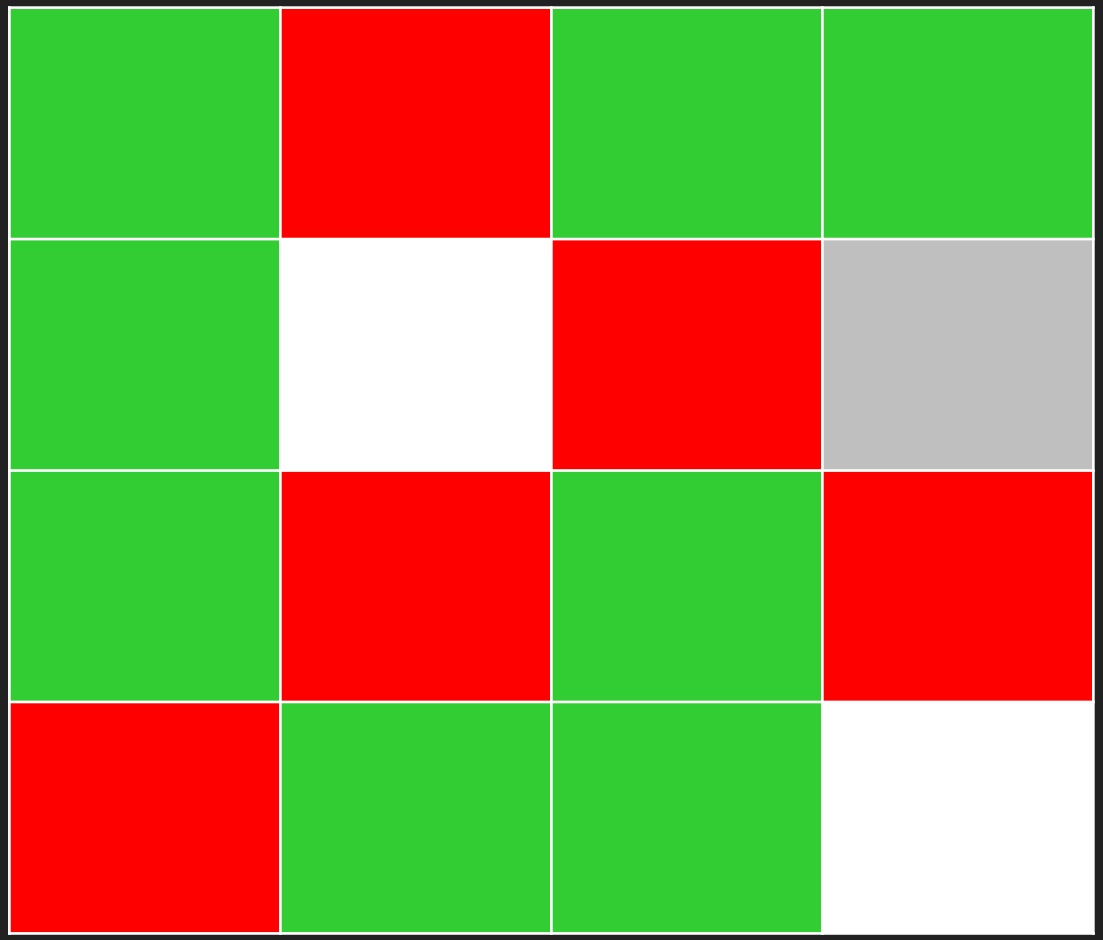
Modélisation numérique

Informatique Pratique

Principe du modèle



Principe du modèle



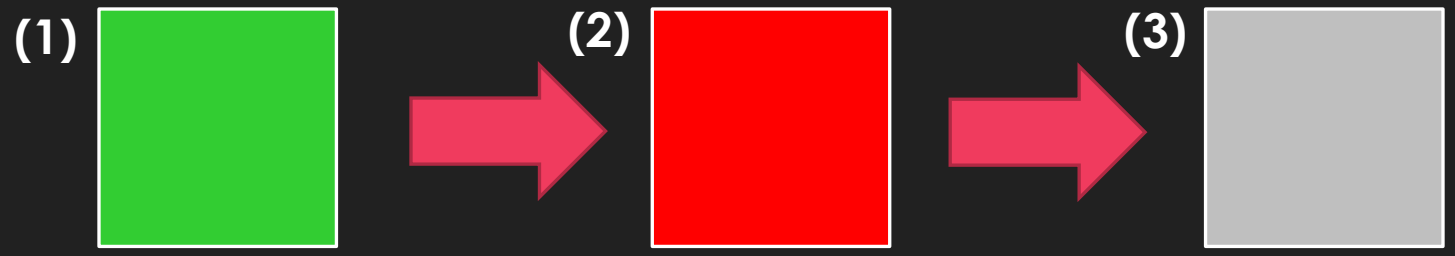
Arbre sain

Arbre consumé

Arbre enflammé

Pas d'arbre

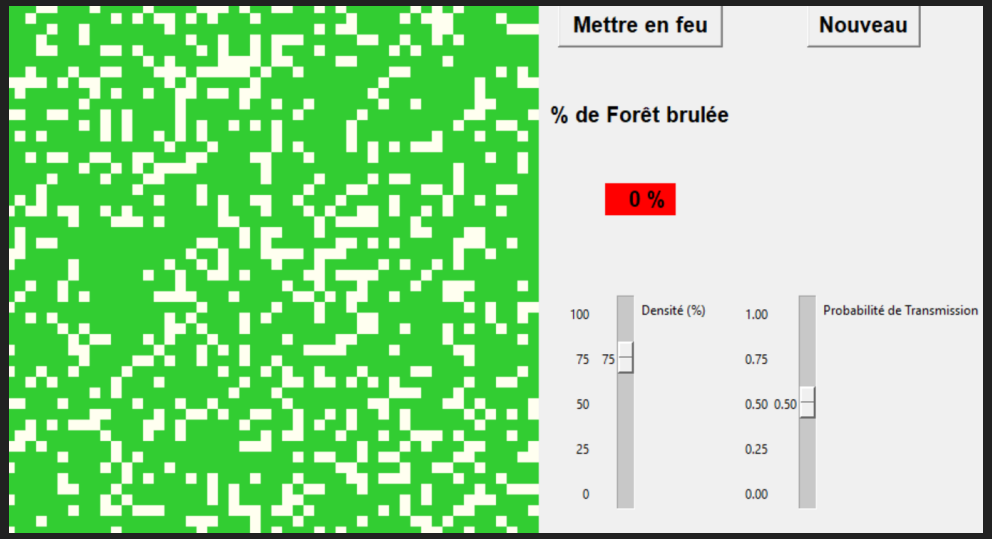
Principe du modèle



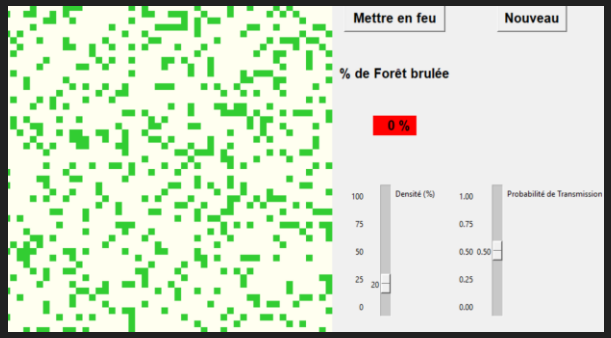
L'arbre sain (1) s'enflamme (2) par son voisin direct selon la probabilité de transmission, puis il se consume (3)

Principe du modèle

Densité moyenne

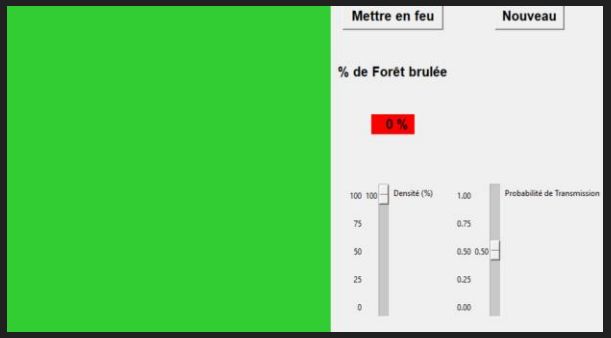


Densité faible

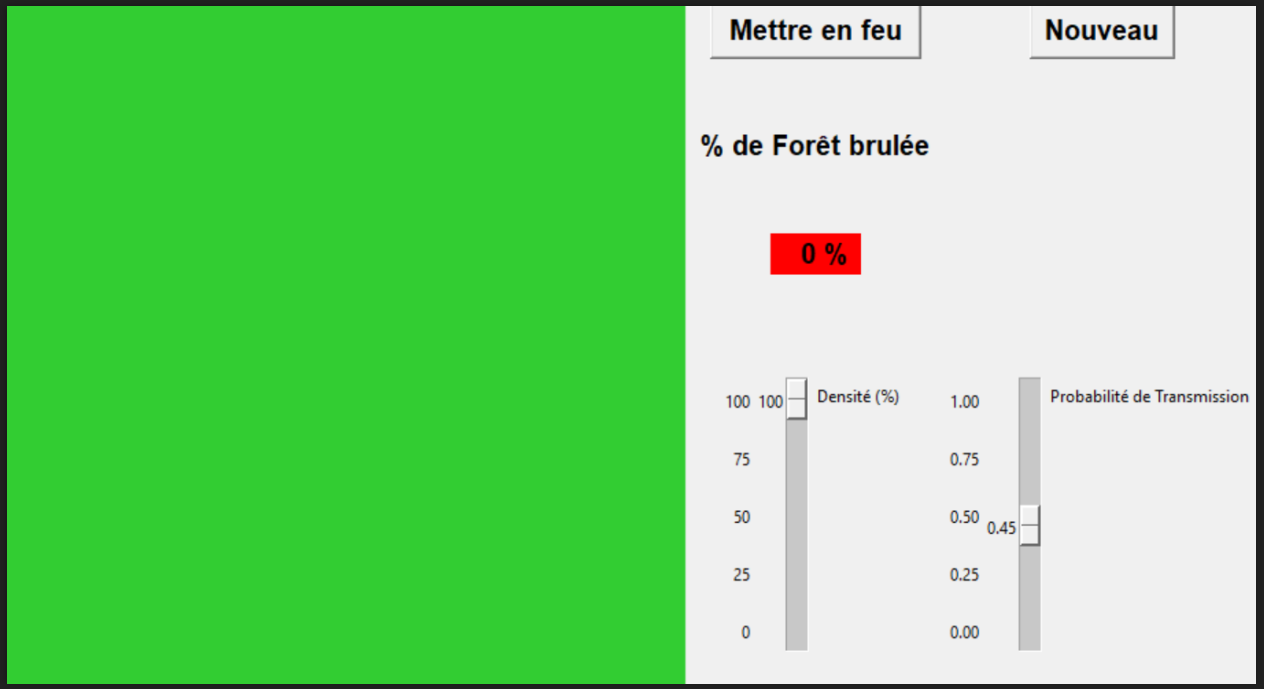


Code 1 en annexe

Densité élevée



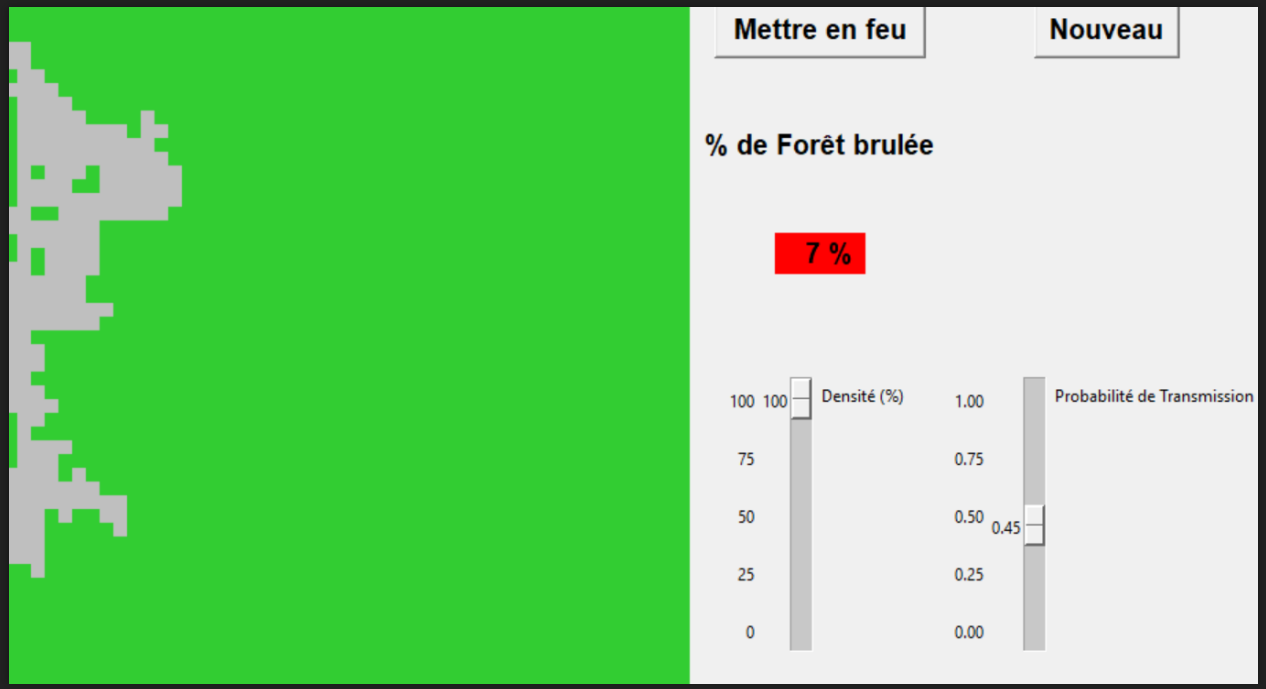
Mise en évidence d'un phénomène de lien



Probabilité de transmission = 0,45

Densité (%) = 100 %

Mise en évidence d'un phénomène de lien

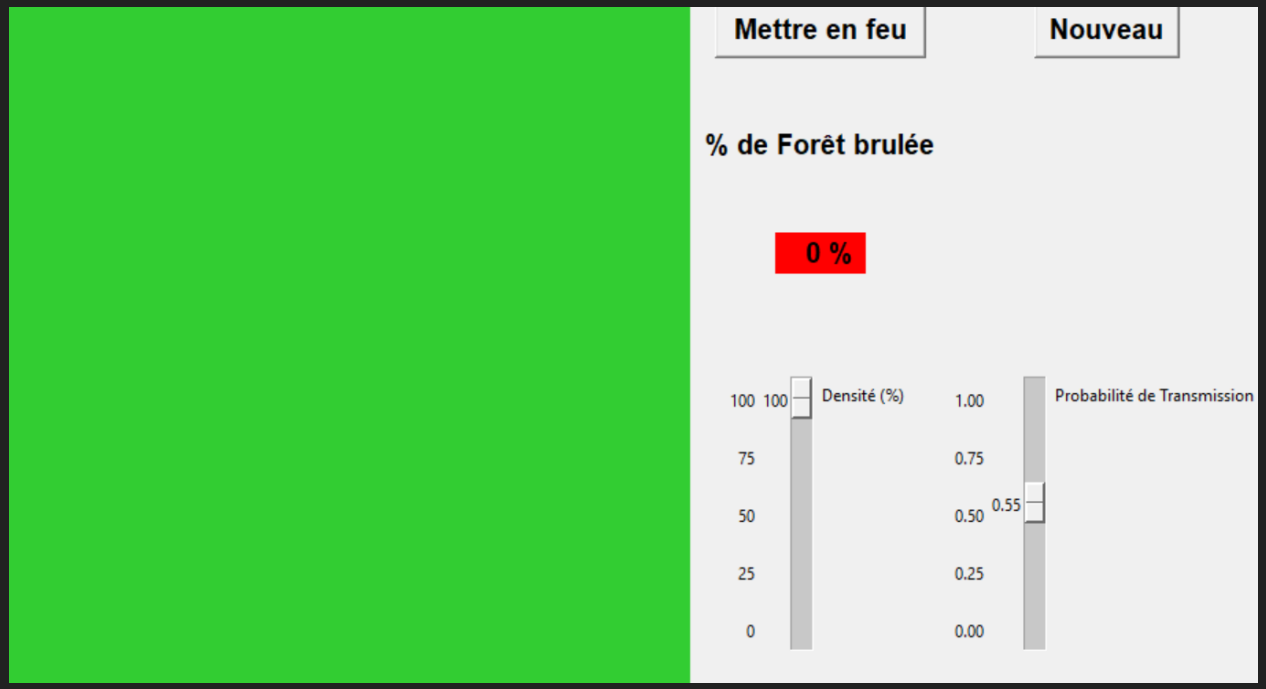


Probabilité de transmission = 0,45

Densité (%) = 100 %

7% de forêt brulée

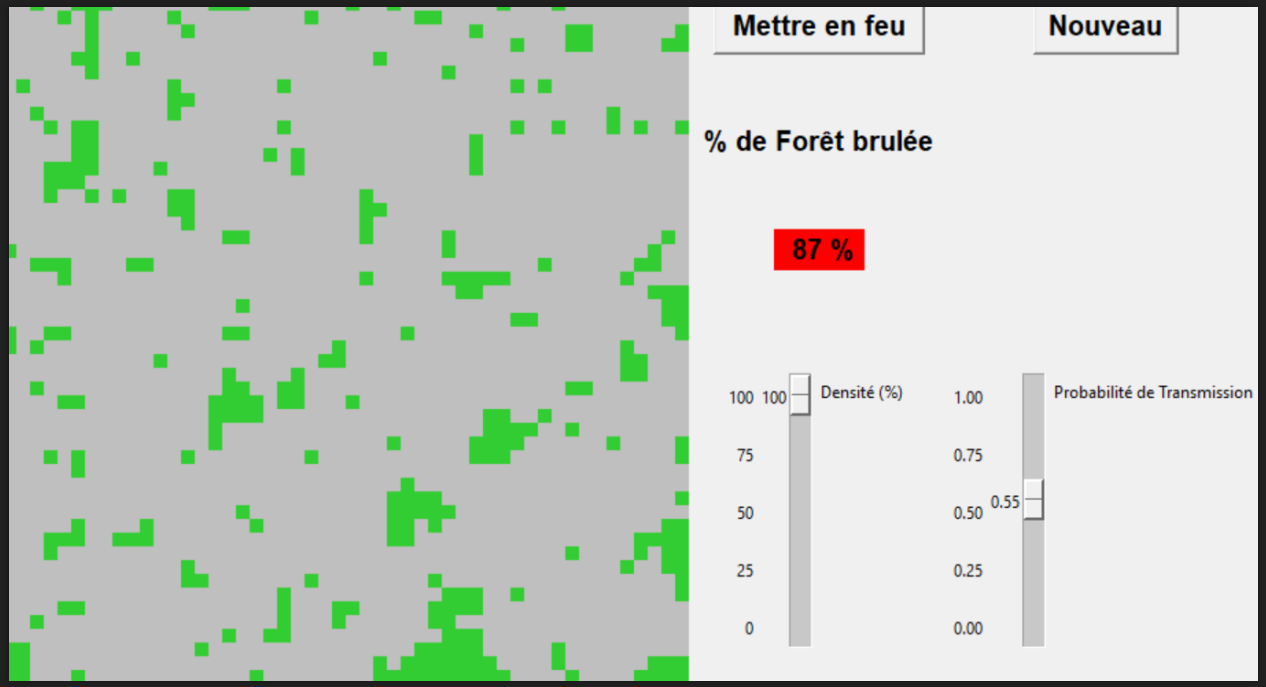
Mise en évidence d'un phénomène de lien



Probabilité de transmission = 0,55

Densité (%) = 100 %

Mise en évidence d'un phénomène de lien



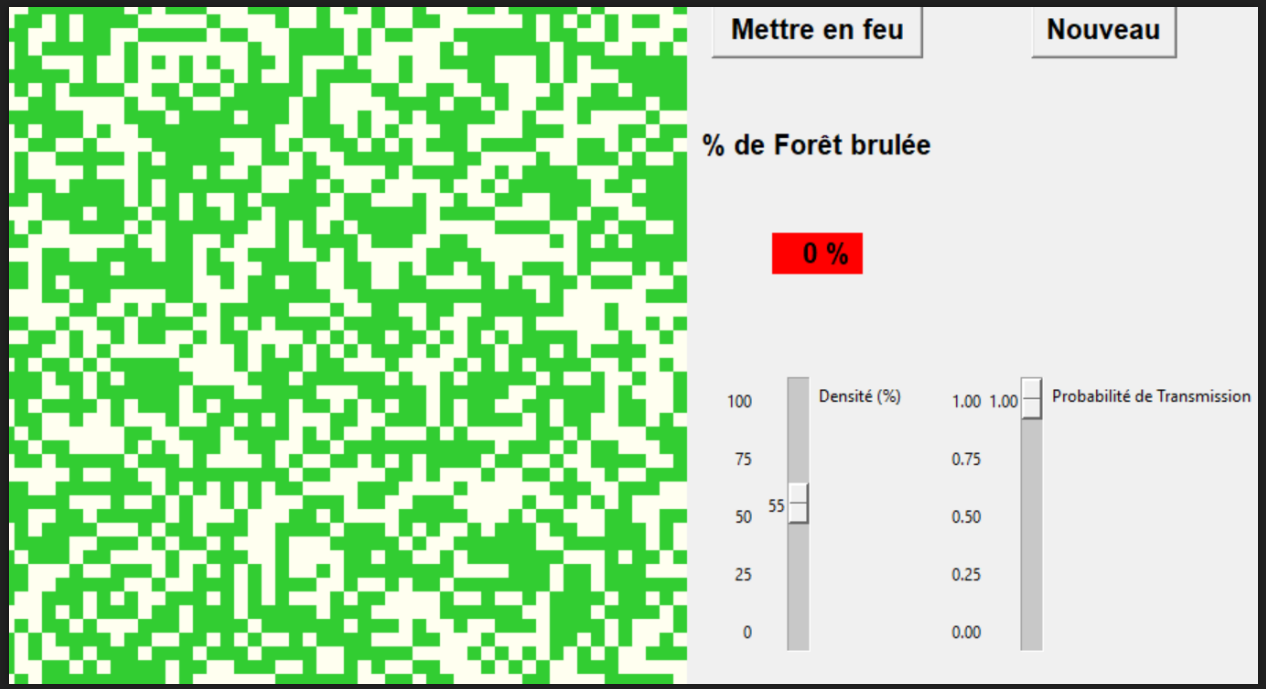
Probabilité de transmission = 0,55

Densité (%) = 100 %

87% de forêt brulée

Un **phénomène** semble se produire dans **l'intervalle de probabilité [0,45 ; 0,55]**

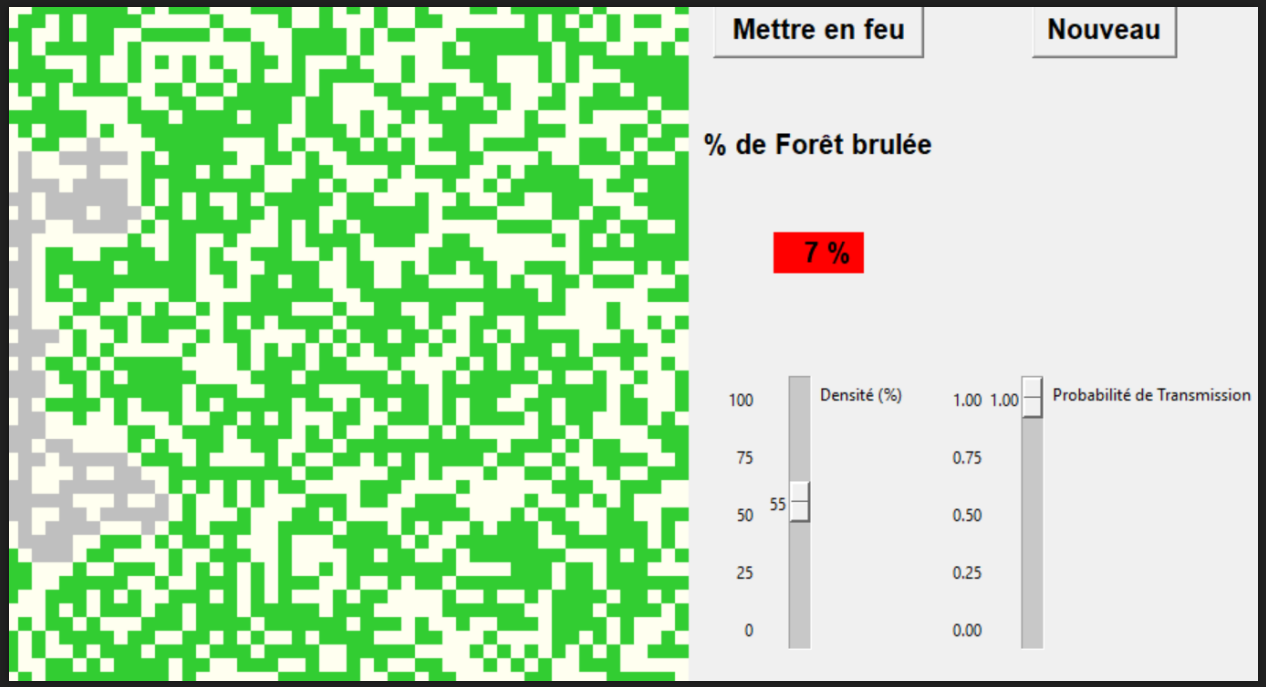
Mise en évidence d'un phénomène de site



Probabilité de transmission = 1

Densité (%) = 55 %

Mise en évidence d'un phénomène de site

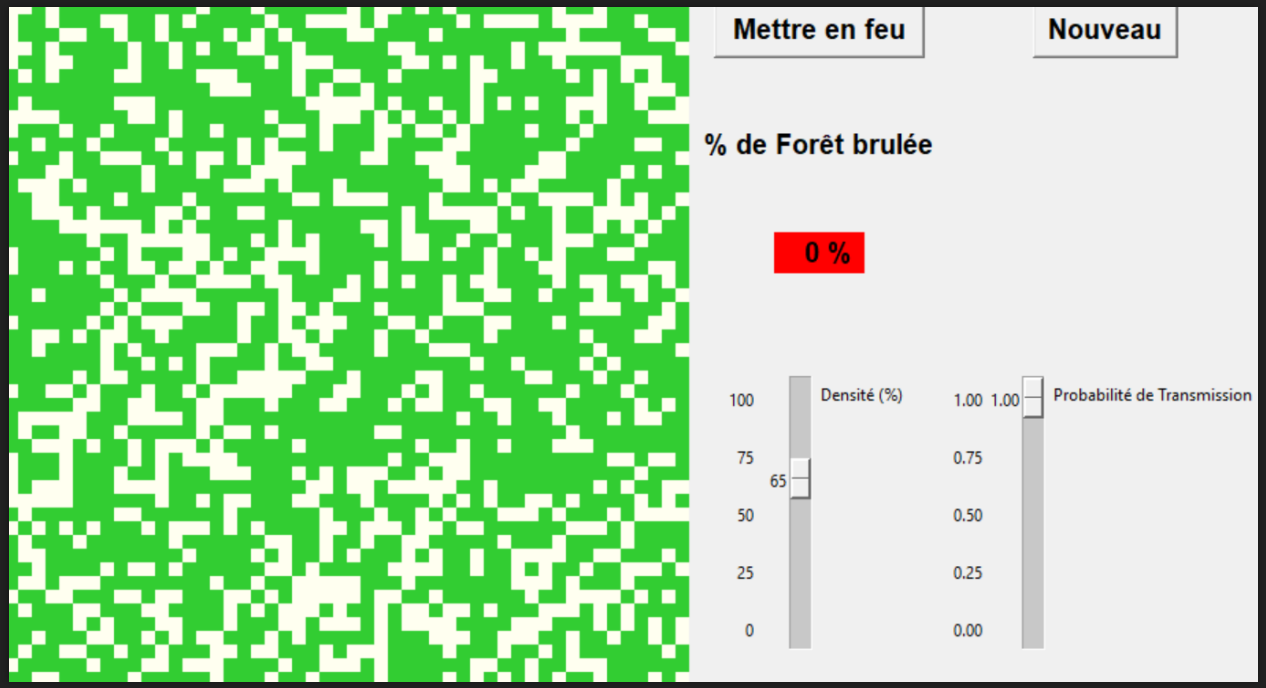


Probabilité de transmission = 1

Densité (%) = 55 %

7% de forêt brulée

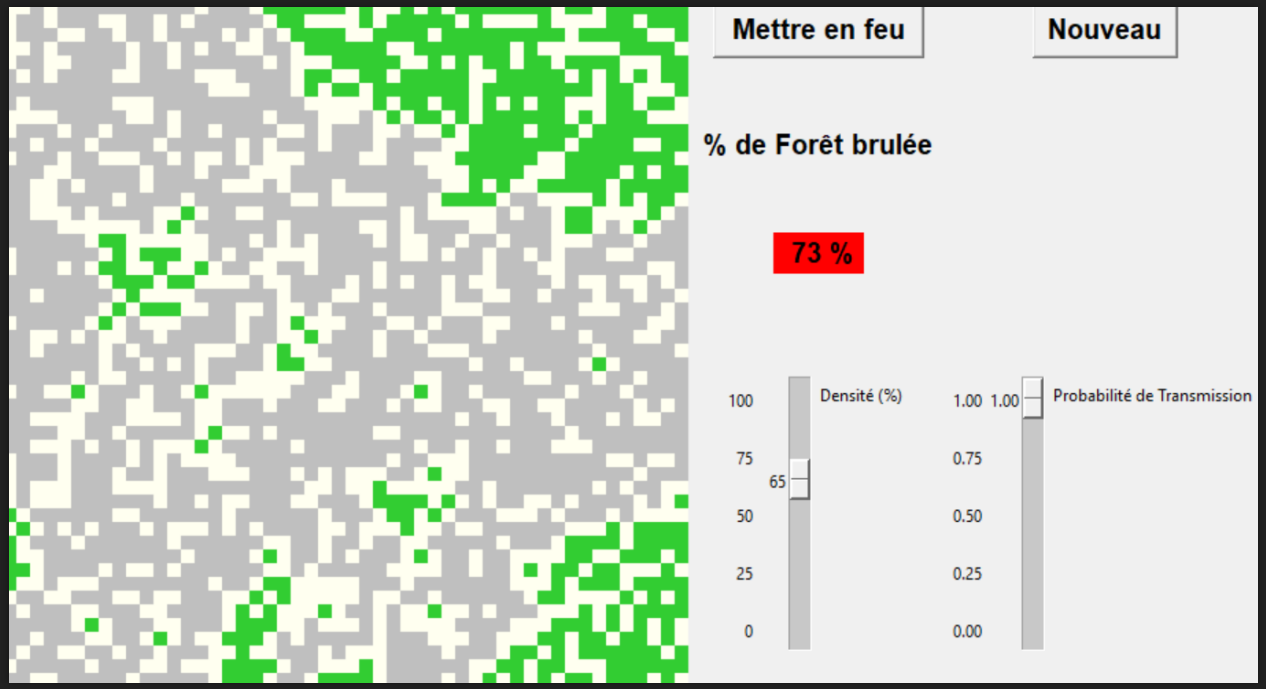
Mise en évidence d'un phénomène de site



Probabilité de transmission = 1

Densité (%) = 65 %

Mise en évidence d'un phénomène de site



Probabilité de transmission = 1

Densité (%) = 65 %

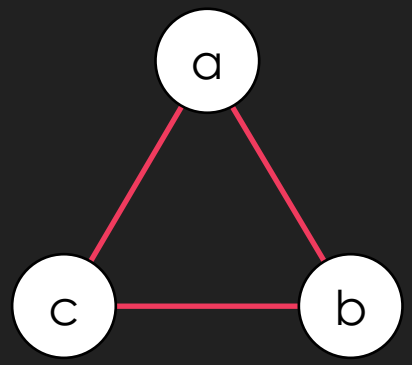
73% de forêt brulée

Un **phénomène** semble se produire dans **l'intervalle de densité [0,55 ; 0,65]**

Explication mathématique par la théorie des graphes

Mathématiques appliquées

Définition | Graphe

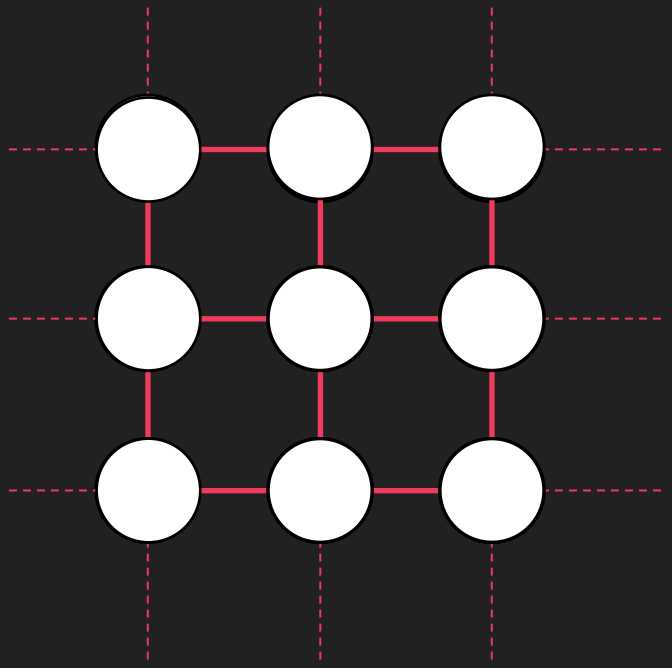


Graphe G = (**V**,**E**)

Où **V** = {sommets de G}
= {a,b,c}

Et **E** = {arrêtes de G, reliant les sommets}
= {(a,b),(b,c),(c,a)}

Définition | Graphe L^2

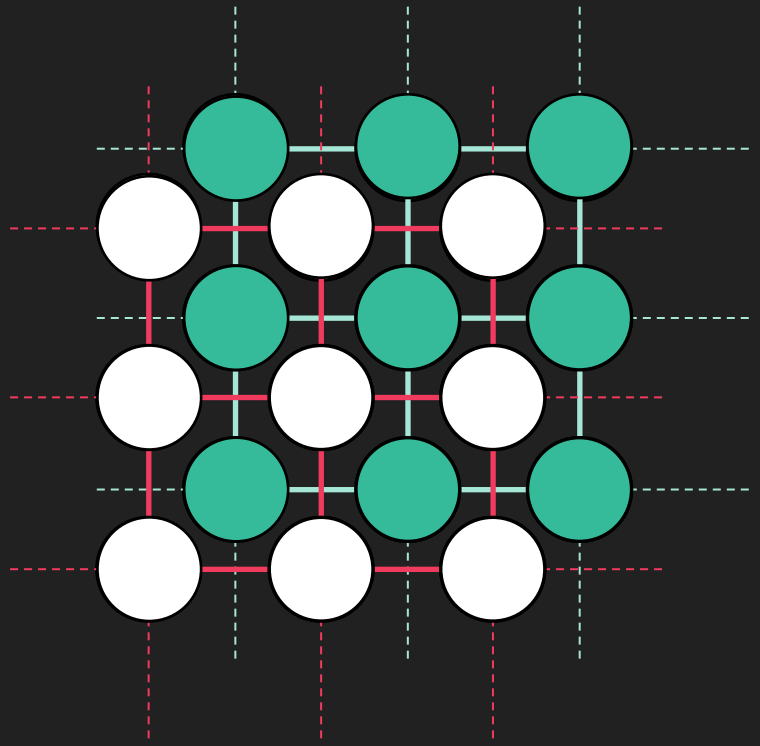


Graphe planaire L^2

Sites (sommets) :
Emplacements possibles des arbres

Liens (arrêtes) :
- Ouverts : propagent le feu
- Fermés : ne propagent pas le feu

Définition | Graphe dual



G* : Graphe dual de G

<u>Construction du dual pour L^2</u> :

Translation de $\begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$ de L^2

Définition | Percolation

On dit qu'il y a **percolation** lorsqu'il y a un **amas infini** :

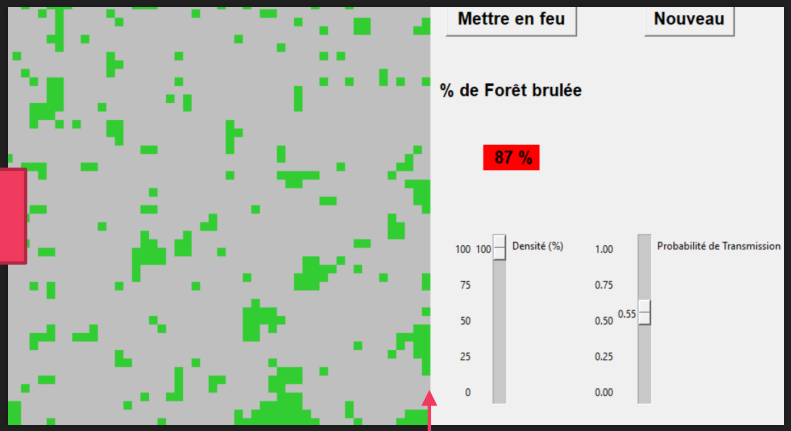
⇒ Ici, lorsque le feu parcourt entièrement la carte.
⇒ Nécessite de simuler sur une grande carte (infini).

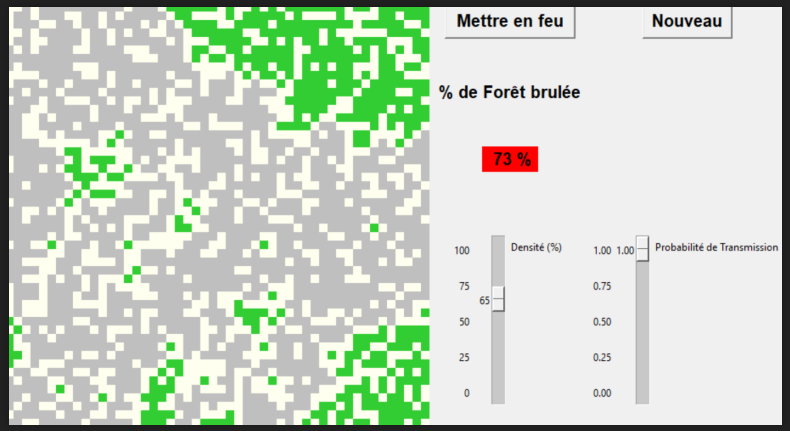
<u>On définit :</u>

- **Chemin** : C(x) = $\{y \in G \ / \ x \leftrightarrow y\}$
- **Probabilité de percolation** : θ_x(p)= P(| C(x) | = ∞)
- **Probabilité critique** : Pc = sup { p / θ_x(p)=0 }

Définition | Percolation

- Percolation de lien
- Percolation de site





Départ du feu

Le feu a atteint le côté droit,
« amas infini » : Il y a **percolation**

Estimation numérique des seuils de percolation | Lien

Sur une forêt de **très grande taille** : 100 x 100

Pour chaque 0,01 de **probabilité de transmission du feu** entre 0 et 1 on effectue 100 simulations, soit 10 000 simulations, qui renvoient :

- 1 si **il y a percolation** (le feu atteint le côté droit)
- 0 si **il n'y a pas percolation** (le feu n'atteint pas le côté droit)

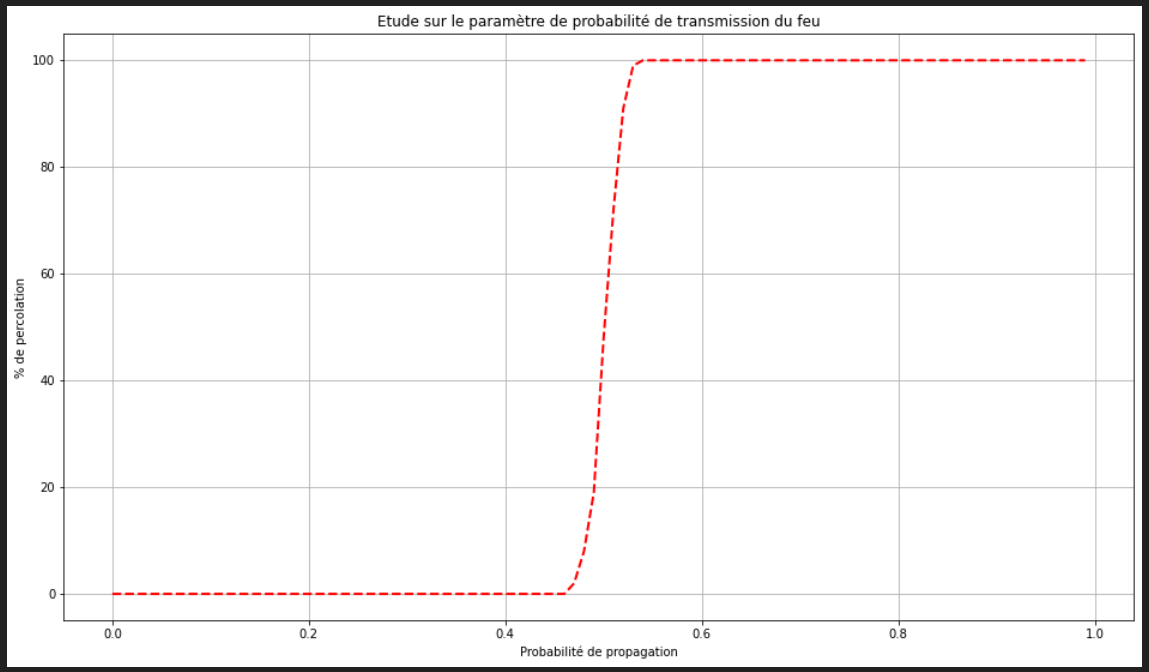
C'est-à-dire la **fonction indicatrice de percolation** :

$$1_{perc}(x) : [0,1] \rightarrow \{0,1\}$$

$$x \rightarrow \begin{cases} 1 & si\ il\ y\ a\ percolation \\ 0 & sinon \end{cases}$$

Code 2 en annexe

Estimation numérique des seuils de percolation | Lien



On trace le **% de percolation** en fonction de la **probabilité de propagation**

= 0,5

Seuil de percolation de lien

Estimation numérique des seuils de percolation | Site

Sur une forêt de **très grande taille** : 100 x 100

Pour chaque 0,01 de **densité** entre 0 et 1 on effectue 100 simulations, soit 10 000 simulations, qui renvoient :

- 1 si **il y a percolation** (le feu atteint le côté droit)
- 0 si **il n'y a pas percolation** (le feu n'atteint pas le côté droit)

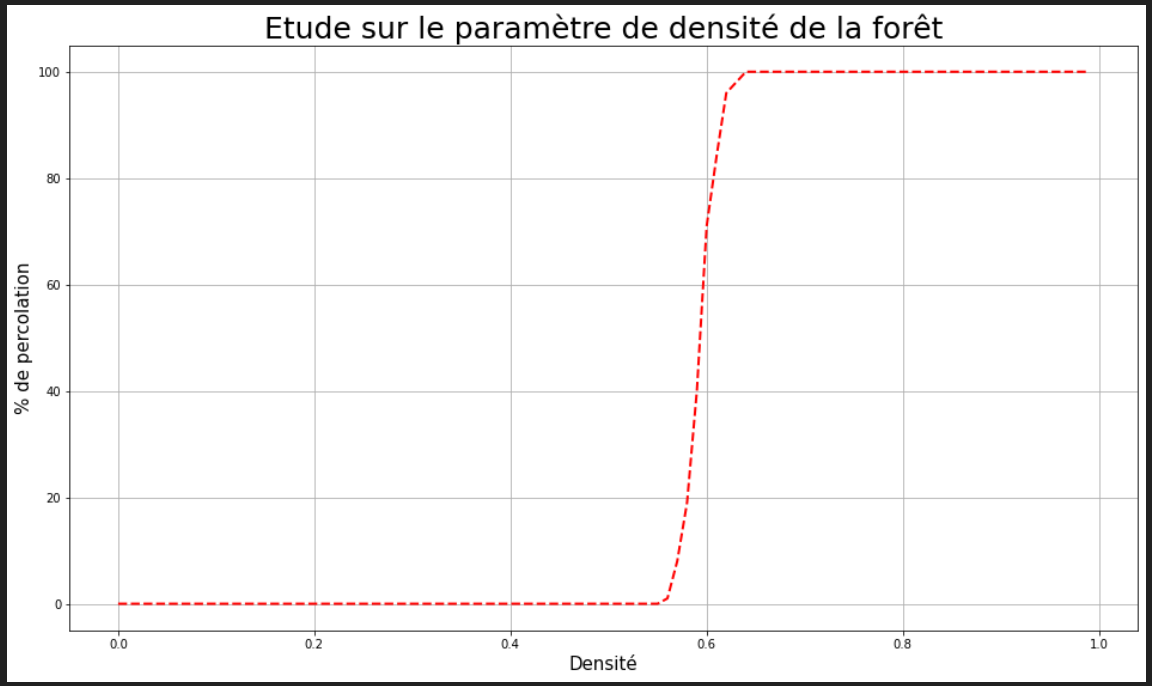
C'est-à-dire la **fonction indicatrice de percolation** :

$$1_{perc}(x) : [0,1] \rightarrow \{0,1\}$$

$$x \rightarrow \begin{cases} 1 & si\ il\ y\ a\ percolation \\ 0 & sinon \end{cases}$$

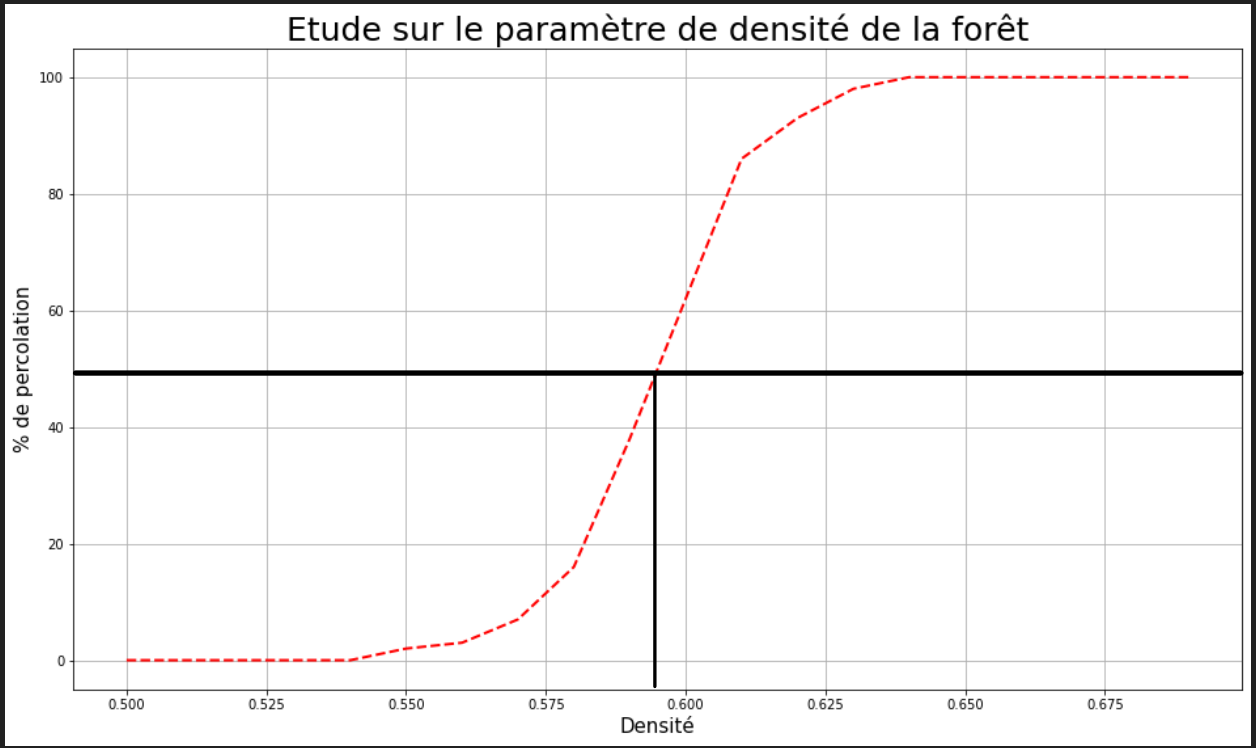
Code 2 en annexe

Estimation numérique des seuils de percolation | Site



On trace le **% de percolation** en fonction de la **densité**

Estimation numérique des seuils de percolation | Site



≈ 0,59

Seuil de percolation de site

Bilan de la modélisation

- Seuil de percolation de lien :

0,5

- Seuil de percolation de site :

0,59

Démonstration du seuil de percolation de lien

Proposition : La somme des seuils de percolation d'un réseau et de son dual vaut 1

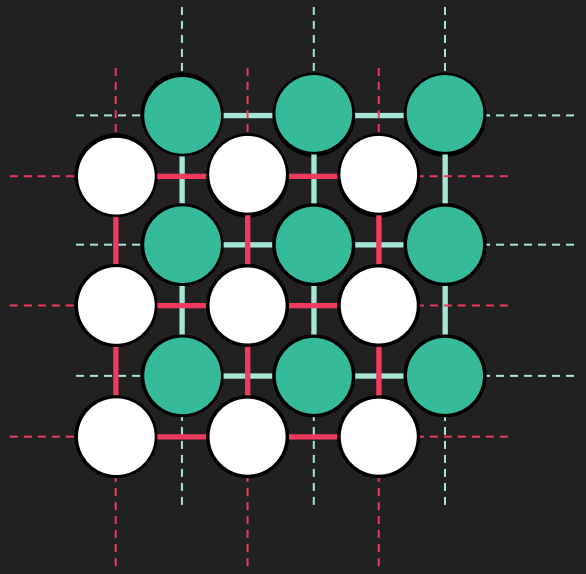
$\mathbf{Pc + Pc_{dual} = 1}$

Or le réseau carré étant son propre dual :

$\mathbf{Pc = Pc_{dual}}$

Donc **2Pc = 1**

i.e. **Pc = ½** (Valeur exacte)



Bilan de la modélisation

- **Seuil de percolation de lien :**

0,5

- **Seuil de percolation de site :**

0,59

Graphe	Site	Lien
$\mathbb{L}^2$	0.592746	0.5
$\mathbb{L}^3$	0.3116	0.2488
$\mathbb{L}^4$	0.197	0.1601
Triangulaire	0.5	0.34729
Diamand	0.43	0.388

Source :
Stauffer D. & Aharony A. (1994) Introduction to percolation theory, revised second edition. CRC Press, Florida.

Validité du modèle et applications

Pertinence du modèle

Validité et vérification

Cycle des végétaux

Continuité

Assèchement

Densité

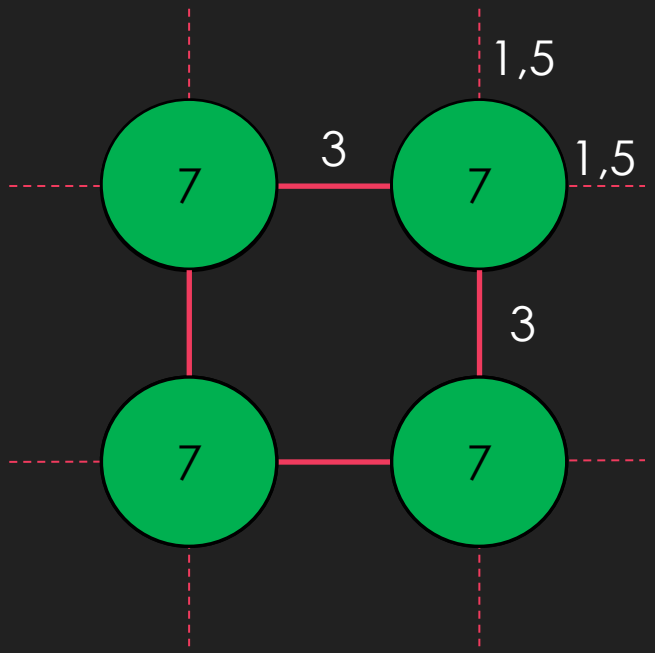
Source : CNRS

Validité et vérification

Pin maritime

Envergure moyenne : 7 m

Validité et vérification



Superficie de la maille : 20^2 = 400 m²

Superficie du pin : $3{,}5^2\pi$ = 38 m²

Superficie des 4 pins : 4x38 = 152 m²

Densité de la forêt : $\frac{152}{400}$ = 0,38 < 0,59

Limites du modèle

Arbres identiques
Forêt homogène

Vent

Relief

Temps

Classification des forêts

- Peuplement (Inflammabilité)
- Densité du couvert végétal

- Relief (Difficulté d'intervention)
- Vent (Accélère et dirige la propagation)
- Présence d'activités humaines

⇒ Surveiller, réglementer, aménager les zones les plus à risque
⇒ Prévenir des feux de forêts

FIN

Annexes

Annexes | Code Graphique

```
"Code Graphique de la modélisation de feu de forêt en fonction de la densité et de la transmission"

#---------- IMPORTATION DES MODULES ----------
from random import sample, randrange, random
from tkinter import Tk, Canvas, Scale, Button, Label, N, ALL

#---------- PARTIE CODE ----------

#Génère une forêt carré aléatoire de densité p contenant n² arbres
def forêt_aléatoire(p, n):
    carré=[(ligne,colonne) for colonne in range(n) for ligne in range(n)]
    nombre_arbres=int(n**2*p)
    arbres=sample(carré,nombre_arbres)
    états=[[0]*n for _ in range(n)]
    for (i,j) in arbres:
        états[i][j]=1
    return états

#Donne les arbres voisins d'un arbre dans les 4 directions G,D,B,H
def voisins(n, i, j):
    return [(a,b) for (a, b) in
            [(i, j+1),(i, j-1), (i-1, j), (i+1,j)]
            if a in range(n) and b in range(n)]

#Permet d'utiliser la probabilité de propagation
def probabilité (prob):
    if random() <= prob:
        return True
    return False

#Rempli la cellule de sa couleur correspondante
def remplir_cellule(états, ligne, colonne):
        A=(unité*colonne, unité*ligne)
        B=(unité*(colonne+1), unité*(ligne+1))
        état=états[ligne][colonne]
        couleur=COULEURS[état]
        cnv.create_rectangle(A, B, fill=couleur, outline='')
```

```
#Rempli la grille à l'aide de la fonction précédente
def remplir(états):
    n=len(états)
    for ligne in range(n):
        for colonne in range(n):
            remplir_cellule(états, ligne, colonne)

#Met à jour la grille de la forêt 1=>2=>3
def mise_à_jour_grille(états):
    n=len(états)
    prendre_feu=[]
    for ligne in range(n):
        for colonne in range(n):
            if états[ligne][colonne]==2:
                états[ligne][colonne]=3
                for (i, j) in voisins(n, ligne, colonne):
                    if états[i][j]==1 and probabilité(prob) == True:
                        prendre_feu.append((i, j))
    for (ligne,colonne) in prendre_feu:
        états[ligne][colonne]=2

#Remet la forêt à 0
def initialisation():
    global états, compteur, nombre_arbres, en_cours

    p=int(curseur1.get())/100
    en_cours=False
    compteur=0
    label1["text"]="%3s %%" %0
    curseur1["state"]='normal'
    curseur2["state"]='normal'
    états=forêt_aléatoire(p, n)
    nombre_arbres=int(n*n*p)
    cnv.delete(ALL)
    remplir(états)
```

Annexes | Code Graphique

```python
#Permet de créer la nouvelle forêt de nouvelle densité
def réglage_densité(états, p):
    n=len(états)
    arbres= [(i,j) for i in range(n) for j in range(n) if états[i][j]==1]
    pas_arbre=[(i,j) for i in range(n) for j in range(n) if états[i][j]!=1]
    nouveaux_arbres=int(n*n*p)
    avant=len(arbres)
    delta=abs(nouveaux_arbres-avant)
    if nouveaux_arbres>=avant:
        for (i, j) in sample(pas_arbre, delta):
            états[i][j]=1
    else:
        for (i, j) in sample(arbres, delta):
            états[i][j]=0

#Construis la forêt
def construire_forêt(pourcentage):
    global nombre_arbres

    cnv.delete("all")
    p=float(pourcentage)/100
    nombre_arbres=int(n*n*p)
    réglage_densité(états,p)
    remplir(états)

#Suis la propagation du feu
def propager():
    global compteur, en_cours

    mise_à_jour_grille(états)
    nombre_feux=sum(états[i][j]==2 for i in range(n) for j in range(n))
    compteur+=nombre_feux
    pourcentage = int(compteur/nombre_arbres*100)
    cnv.delete("all")
    remplir(états)
    label1["text"]="%3s %%" %pourcentage
    if nombre_feux==0:
        en_cours=False
        return
    cnv.after(150, propager)
```

```python
#Met en feu
def feu(événement):
    global en_cours, compteur

    i, j=événement.y//unité, événement.x//unité
    if états[i][j]==1:
        états[i][j]=2
        remplir_cellule(états, i, j)

        compteur+=1
        if not en_cours:
            en_cours=True
            curseur1["state"]='disabled'
            curseur2["state"]='disabled'
            propager()

#Met en feu une partie de la gauche de la forêt
def allumer_feu():
    global en_cours, compteur
    i=n//4
    for a in range(n//2):
        états[i+a][1]=2

    compteur+=1
    if not en_cours:
        en_cours=True
        curseur1["state"]='disabled'
        curseur2["state"]='disabled'
        propager()

#Met à jour la probabilité de transmission
def probabilité_curseur(prob_curseur):
    global prob
    prob=float(prob_curseur)


#---------- PARTIE REGLAGE ----------
n=75 #Taille de la forêt (un côté)
unité=10 #Dimension graphique d'un carré sur l'écran
COULEURS=["ivory", "lime green", "red", "gray75"] #Couleurs de la modélisation
```

Annexes | Code Graphique

```
#---------- PARTIE GRAPHIQUE ----------

# Fenêtre et canevas
root = Tk()
cnv = Canvas(root, width=unité*n-2, height=unité*n-2, background="ivory")
cnv.grid(row=0, column=0, rowspan=4)

#Boutons
btn1=Button(root,text="Nouveau",  font='Arial 15 bold', command=initialisation, width=8)
btn1.grid(row=0, column=2, sticky=N)

bouton2=Button(root,text="Mettre en feu",  font='Arial 15 bold', command=allumer_feu, width=12)
bouton2.grid(row=0, column=1, sticky=N)

#Labels
label1=Label(root,text="%3s %%" %0,  font='Arial 15 bold', bg='red', width=5)
label1.grid(row=2, column=1, sticky=N)

label2=Label(root,text="% de Forêt brulée",  font='Arial 15 bold', width=15)
label2.grid(row=1, column=1, sticky=N)

# Mettre le feu avec un clic
cnv.bind("<Button-1>", feu)

#Curseurs
curseur1 = Scale(root, orient = "vertical", command=construire_forêt, from_=100,
      to=0, length=200, tickinterval= 25,  label='Densité (%)')
curseur1.set(50)
curseur1.grid(row=3, column=1)

curseur2 = Scale(root, orient='vertical', command=probabilité_curseur, from_=1, to=0,
      resolution=0.01, length=200, tickinterval= 0.25,
      label='Probabilité de Transmission')
curseur2.set(0.5)
curseur2.grid(row=3, column=2)

initialisation()

root.mainloop()
```

Annexes | Code Statistique

```
"Code Statitique de la modélisation de feu de forêt en fonction de la densité et de la transmission"

#---------- IMPORTATION DES MODULES ----------
from random import sample, randrange, random
import matplotlib.pyplot as plt
import time

#---------- PARTIE CODE ----------

#Génère une forêt carré aléatoire de densité p contenant n² arbres
def forêt_aléatoire(p, n):
    carré=[(ligne,colonne) for colonne in range(n) for ligne in range(n)]
    nombre_arbres=int(n**2*p)
    arbres=sample(carré,nombre_arbres)
    forêt=[[0]*n for _ in range(n)]
    for (i,j) in arbres:
        forêt[i][j]=1
    return forêt

#Donne les arbres voisins d'un arbre dans les 4 directions G,D,B,H
def voisins(n, i, j):
    return [(a,b) for (a, b) in
            [(i, j+1),(i, j-1), (i-1, j), (i+1,j)]
            if a in range(n) and b in range(n)]

#Permet d'utiliser la probabilité de transmission
def probabilité (prob):
    if random() <= prob:
        return True
    return False
```

```
#Met à jour la grille de la forêt 1=>2=>3
def mise_à_jour_grille(forêt, probabilité_propagation):
    prendre_feu=[]
    for ligne in range(n):
        for colonne in range(n):
            if forêt[ligne][colonne]==2:
                forêt[ligne][colonne]=3
                for (i, j) in voisins(n, ligne, colonne):
                    if forêt[i][j]==1 and probabilité(probabilité_propagation) == True:
                        prendre_feu.append((i, j))
    for (ligne,colonne) in prendre_feu:
        forêt[ligne][colonne]=2
    return forêt


#Met en feu lune partie de la gauche de la forêt
def allumer_feu(forêt):
    i=n//4
    for a in range(n//2):
        forêt[i+a][1]=2
    return forêt

#Indique si il y a percolation i.e. amas infini
def percolation (forêt):
    for i in range (n):
        if forêt[i][n-1]==3:
            return True
    return False

#Simule un feu jusqu'à extinction
def simulation(densité, probabilité_propagation):
    forêt = forêt_aléatoire(densité, n)
    forêt = allumer_feu(forêt)
    nombre_feux=sum(forêt[i][j]==2 for i in range(n) for j in range(n))
    while nombre_feux !=0 :
        forêt = mise_à_jour_grille(forêt,probabilité_propagation)
        nombre_feux=sum(forêt[i][j]==2 for i in range(n) for j in range(n))
    if percolation(forêt)==True:
        return 1
    return 0
```

Annexes | Code Statistique

```
#---------- PARTIE ACQUISITION ----------
def acquisition_probabilité_propagation(densité):
    début=time.time()
    L=[]
    P=[]
    probabilité_propagation=0
    while probabilité_propagation<=1:
        P.append(probabilité_propagation)
        s=0
        for _ in range(100):
            s+=simulation(densité, probabilité_propagation)
        print((int(probabilité_propagation*100)),"/100")
        L.append(s)
        probabilité_propagation+=0.01
    fin=time.time()
    print("Temps de calcul =",fin-début)
    plt.figure(figsize=(16,9))
    plt.plot(P,L, color = 'red', linestyle = '--' , linewidth = 2)
    plt.xlabel("Probabilité de propagation")
    plt.ylabel("% de percolation")
    plt.grid()
    plt.title("Etude sur le paramètre de probabilité de transmission du feu")
```

Annexes | Code Statistique

```
def acquisition_probabilité_propagation_centrée():
    début=time.time()
    L=[]
    P=[]
    probabilité_propagation=0.4
    while probabilité_propagation<=0.6:
        P.append(probabilité_propagation)
        s=0
        for _ in range(100):
            s+=simulation(1, probabilité_propagation)
        print((int(probabilité_propagation*100)-40),"/20")
        L.append(s)
        probabilité_propagation+=0.01
    fin=time.time()
    print("Temps de calcul =",fin-début)
    plt.figure(figsize=(16,9))
    plt.plot(P,L, color = 'red', linestyle = '--' , linewidth = 2)
    plt.xlabel("Probabilité de propagation")
    plt.ylabel("% de percolation")
    plt.grid()
    plt.title("Etude sur le paramètre de probabilité de transmission du feu")
```

Annexes | Code Statistique

```
def acquisition_densité(probabilité):
    début=time.time()
    L=[]
    D=[]
    densité=0
    while densité<=1:
        D.append(densité)
        s=0
        for _ in range(100):
            s+=simulation(densité, probabilité)
        print((int(densité*100)),"/100")
        L.append(s)
        densité+=0.01
    fin=time.time()
    print("Temps de calcul =",fin-début)
    plt.figure(figsize=(16,9))
    plt.plot(D,L, color = 'red', linestyle = '--' , linewidth = 2)
    plt.xlabel("Densité", fontsize = 15)
    plt.ylabel("% de percolation", fontsize = 15)
    plt.grid()
    plt.title("Etude sur le paramètre de densité de la forêt", fontsize = 25)
```

Annexes | Code Statistique

```
def acquisition_densité_centrée():
    début=time.time()
    L=[]
    D=[]
    densité=0.5
    while densité<=0.7:
        D.append(densité)
        s=0
        for _ in range(100):
            s+=simulation(densité, 1)
        print((int(densité*100)-50),"/20")
        L.append(s)
        densité+=0.01
    fin=time.time()
    print("Temps de calcul =",fin-début)
    plt.figure(figsize=(16,9))
    plt.plot(D,L, color = 'red', linestyle = '--' , linewidth = 2)
    plt.xlabel("Densité", fontsize = 15)
    plt.ylabel("% de percolation", fontsize = 15)
    plt.grid()
    plt.title("Etude sur le paramètre de densité de la forêt", fontsize = 25)
```

Annexes | Code Statistique

```
def acquisition_densité_précision(epsilon):
    début=time.time()
    densité_min=0.59
    densité_max=0.60
    densité_pivot=(densité_max+densité_min)/2
    while densité_max-densité_min > epsilon:
        densité_pivot=(densité_max+densité_min)/2
        print(densité_pivot)
        s=0
        for _ in range(100):
            s+=simulation(densité_pivot, 1)
        if s>50:
            densité_max=densité_pivot
        elif s==50:
            fin=time.time()
            print("Temps de calcul =",fin-début)
            return densité_pivot
        else:
            densité_min=densité_pivot
    fin=time.time()
    print("Temps de calcul =",fin-début)
    return densité_pivot

#---------- PARTIE REGLAGE ----------
n=100 #Taille de la forêt (un côté)
COULEURS=["ivory", "lime green", "red", "gray75"] #Couleurs de la modélisation
```