Le 1er est un programme avec lequel nous avons créé un film.

```
#-*-coding: utf-8-*-
#########################################################
# Petit module pour la percolation d'un feu de foret #
#########################################################
# Pour la foret :
# 0-vert 1-enflamme´ 2-Carbonise´
# pavage carre´
#
#
# A -> la matrice repre´sentant la foret
# R -> la liste des zones enflame´es

from numpy import *
from random import *
from PIL import Image

####################
# La partie Matrice #
####################

def carre_vide(n):
        # "Retourne une matrice vide carre´ de dimension n"
        A=zeros((n,n), int)
        return A

def border(A,n):
        # "Ajoute un tour carbonise´ a` A"
        ligne=2 + zeros((1 ,n) , int)
        A=concatenate((A,ligne))
        A=concatenate((ligne ,A))
        colonne = 2 + zeros((n+2, 1), int)
        A=concatenate((A, colonne) ,1)
        A=concatenate((colonne ,A) ,1)
        return A

#########################################################
# La partie des proce´dures ne´cessaires a` l'algorithme #
#########################################################

#initialisation

def deboiser(A,n,p):
        # "De´boise au hasard avec une proba de p pour chaque zone"
        for i in range (1 ,n+1):
                for j in range (1 ,n+1):
                        if (random()<p):
                                A[i,j]=2
        return A

def enflamer(A,n):
        # " Allume      une     zone au hasard "
        x=randint (1 ,n)
        y=randint (1 ,n)
        R=[]
        R.append ([ x , y ])
        A[x,y]=1
        return A,R


def enflamer_centre(A,n):
        # "Allume       la      zone    au      centre "
        A[n/2,n/2]=1
        R=[]
        R.append([n/2,n/2])
        return A,R

def enflamer_front(A,n):
        # "Allume un front de feu au nord"
        R=[]
```

```
        for i in range (1,n+1):
                A[1,i]=1
                R.append([1,i])
        return A,R

#Une etape 4 directions pavage carre´
def une_etape(A,R,n,p):
        #"calcul pour une e´tape en sortie avec une proba p d'inflamation"
        B=copy(A)
        S=[]
        for feu in R:
                i,j=feu[0],feu[1]
                for k in range (-1,1):
                        if ((random()<p) and B[i+k,j]==0):
                                B[i+k,j]=1
                                S.append([i+k,j])
                        if ((random()<p) and B[i,j+k]==0):
                                B[i,j+k]=1
                                S.append([i,j+k])
                B[i,j]=2
        A=copy(B)
        return A,S


#Une e´tape 8 directions pavage carre´
def une_etape_carre_8(A,R,n,p):
        #"calcul pour une e´tape en sortie avec une proba p d'inflamation 8 directions"
        B=copy(A)
        S=[]
        for feu in R:
                i,j=feu[0],feu[1]
                for k in range (-1,2):
                        for l in range (-1 ,2):
                                if ((k!=0 or l!=0)):
                                        if ((random()<p) and B[i+k,j+l]==0):
                                                B[i+k,j+l]=1
                                                S.append([i+k,j+l])
                B[i,j]=2
        A=copy(B)
        return A,S


def compte_zone (A, n , k ) :
        #"Compte le nombre de zone de A qui contient la valeur k"
        conteur=0
        for     i in range(1,n+1):
                for     j in range(1,n+1):
                        if (A[i,j]==k):
                                conteur+=1 #incremente conteur
        return conteur


def compte_frontiere_atteinte ( liste , longueur ):
        #"Compte dans la liste de re´ponse de longueur longueur le nombre de fois ou` la frontiere est
atteinte avant la fin"
        resultat=0
        for     i in range(longueur):
                if (liste[i][1]==1):
                        resultat+=1
        return resultat


def feu_sur_frontiere(A,n):
        #"retourne 1 si un point de la frontie`re est en feu"
        reponse=0
        for     i in range (1,n+1):
                if ((A[i,1]==1) or (A[i,n]==1) or (A[1,i]==1) or (A[n,i]==1)):
                        reponse=1
        return reponse


################################
```

```
# La Partie Traitement d'Image #
################################

def obtention_image(A,n,nom_de_fichier):
        # "Donne l'image de la foret a` partir de la matrice : fichier png"
        # valeurs pour points foret , enflame´ , carbonise´
        colors  =       [(128 ,0 ,0) ,(0 ,0 ,255) ,(48 ,48 ,48)]
        # conversion en chaine de 3 caracteres
        colors = [ ''.join([chr(x) for x in color]) for color in colors ]
        # construction d'une chaine avec des valeurs issue de A par pixel
        # pour une image de taille nxn
        img_str =       ' '
        for line in range(n):
                for col in range(n):
                        img_str +=      colors [A[ line , col ]]
        # creation de l 'image d'apres cette chaine
        img = Image.fromstring('RGB' ,(n,n),img_str)
        # sauvegarde au format PNG
        img.save(nom_de_fichier,'PNG')
        return True


############################
# La partie nom de fichier #
############################

def nom_image(n,base_nom):
        #"Fonction pour faciliter le montage vide´o"
        Nom=base_nom + (len(base_nom) - len(str(n))) * '0' + str(n) + '.png'
        return Nom


def Nom_de_fichier_csv(n,proba):
        #"Cre´e un nom de fichier"
        Nom= '100_tirages_sur_carte_ ' + str(n) + ' x ' + str(n) + ' _ avec_proba_de_ ' + str(proba) +
'.csv'
        return Nom


def ZEF(n,proba,nb_de_cas):
        #"Cre´e le nom du fichier csv pour l'e´tude des zones en feu"
        Nom=str(nb_de_cas) + ' _etude_de_zones_en_feu_avec_proba_de_ ' + str(proba) + '.csv'
        return Nom


##################
# Les Programmes #
##################

def Film(n,p, carte ):
        #"Obtention des images pour une carte nxn et une proba p"
        A=carre_vide(n)
        A=border(A,n)
        #A,R=enflamer_centre(A,n) # version originelle du code source
        A,R=enflamer(A,n)
        print(A)
        print(R)
        i=-1 #compteur pour suivre la chronologie
        while (R!=[]):
                i +=1
                if carte==0:
                        A,R=une_etape(A,R,n,p)
                        print "e´tape :",i," avec ",compte_zone(A,n,1)," zones en feu et R : ",R
                        obtention_image(A,n+2,nom_image(i ,'Foret'))
                if carte==1:
                        A,R=une_etape_carre_8(A,R,n,p)
                        print "e´tape :",i," avec ",compte_zone(A,n,1)," zones en feu et R : ",R
                        obtention_image(A,n+2,nom_image(i ,'Foret'))
        return A


def Zones_en_feu(n,p,carte ,nb_de_cas):
```

```
        # "Etude des zones en feu a` chaque e´tape. Ecriture des re´sultats dans un fichier"
        f = open(ZEF(n,p,nb_de_cas), "w")
        La_liste =[] # Une liste de liste - chaque sous liste contient le nombre de zones en feu a`
chaque e´tape
        Liste_temp_premiere_brulure=[] #Liste ou` se trouve l'e´tape de premier contacte avec la
frontie`re
        for i in range (nb_de_cas):
                A=carre_vide(n)
                A=border(A,n)
                A,R=enflamer_centre (A,n)
                liste_local =[1] # La fameuse sous liste
                k=1 #compteur temps pour suivre la chronologie et donner le premier moment ou` la
frontie`re est atteinte
                bordbrule=0 #0 si le bord n'a jamais brule´ . S' il a e´te´ atteint bordbrule vaut le
temps (k)
                while (R!=[]):
                        if carte==0:
                                k+=1
                                A,R=une_etape(A,R,n,p)
                                if (( bordbrule==0)     and     ( feu_sur_frontiere (A,n)==1)):
                                        bordbrule=k
                                liste_local.append(compte_zone(A,n,1))
                La_liste.append(liste_local)
                Liste_temp_premiere_brulure.append(bordbrule)
        ### Partie pour la cre´ation du fichier csv
        longueurs =[]
        for     i in La_liste :
                longueurs.append(len(i))
        longueur_max=max(longueurs)
        for i in range(len(La_liste)): # Dans cette boucle la liste est comple´te´e avec des ze´ros
pour e´crire facilement le fichier csv
                for     j in range(longueur_max - len(La_liste[i])):
                        La_liste[i ].append(0)
        ligne=' '
        for j in range(len(La_liste )): #Dans cette boucle on ecrit les temps de brulure de la
frontie`re
                ligne=ligne + ' k= ' + str(Liste_temp_premiere_brulure[j]) + ' '
        ligne=ligne+" \n"
        f.write(ligne)
        for i in range(longueur_max):
                ligne=' '
                for j in range(len(La_liste)):
                        ligne=ligne+str(La_liste[j][i])+' '
                ligne=ligne+" \n"
                f.write(ligne)
        f.close()
        return La_liste


def Etude_100_tirages(n,p, carte ):
        # "100 tirage sur carte nxn avec proba p en sortie une liste du nb de carbonise´ et 1 si
frontiere atteinte"
        liste_resultat=[]
        for     i in range (100):
                A=carre_vide(n)
                A=border(A,n)
                A,R=enflamer_centre(A,n)
                touche_frontiere=0 #touche_frontiere=(0,1)=(frontiere non atteinte ,frontiere
atteinte)
                while (R!=[]):
                        if (touche_frontiere ==0):
                                touche_frontiere=feu_sur_frontiere(A,n)
                        if (carte==0):
                                A,R=une_etape(A,R,n,p)
                        if (carte==1):
                                A,R=une_etape_carre_8(A,R,n,p)
                liste_resultat.append([compte_zone(A,n,2),touche_frontiere])
        return liste_resultat


def variation_proba(n,depart ,pas,nb_iteration ,carte):
        # "variation de la proba en partant de depart en avancant de pas nb_iterartion fois"
```

```
        frontiere_touche =[]
        for i in range(nb_iteration):
                liste=Etude_100_tirages(n,depart+i*pas,carte)
                print(liste)
                print("Attention la frontiere a ete atteinte ",compte_frontiere_atteinte(liste ,100)," 
fois")
                frontiere_touche.append([depart+i*pas,compte_frontiere_atteinte(liste,100)])
                f=open(Nom_de_fichier_csv(n,depart+i*pas),"w")
                f.write("Nombre de zones carbonne a` la fin\n")
                f.write("avec proba de :"+str(depart+i*pas)+"\n")
                for     j in range (100):
                        f.write(str(liste[j][0])+"\n")
                f.close()
                print(sum(liste)*1.0/(n*n)," % de foret brule pour une proba de ", depart+i*pas)
                print(frontiere_touche)
        return True


def Film_Front(n,p):
        # "Obtention des images pour un front de feu allume´ en Ouest"
        A=carre_vide (n)
        A=border(A,n)
        A=deboiser(A,n,p)
        A,R=enflamer_front(A,n)
        i=-1 #compteur pour suivre la chronologie
        while (R!=[]):
                i+=1
                #A,R=une_etape(A,R,n,1) #ligne originelle du code source
                A,R=une_etape(A,R,n,p)
                print "e´tape :",i," avec ",compte_zone(A,n,1)," zones en feu"
                obtention_image(A,n+2,nom_image(i,'Front'))
        return A


Film(750,0.5,1)

#Film_Front(128,0.5)
#Film_Front(128,variation_proba(128,0.2,0.01,100,1))
```