TIPE: Modélisation des feux de forêts

Gaël Kandel - 16269

Épreuve de TIPE

Session 2023

Plan de l'exposé

Introduction

Les feux de forêt : un danger pour les villes

- Évacuation des habitants

Introduction

Les feux de forêt : un danger pour les villes

- Évacuation des habitants
- Dégâts matériels

Introduction

Les feux de forêt : un danger pour les villes

- Évacuation des habitants
- Dégâts matériels
- Pollution de l'air

Introduction

Les feux de forêt : un danger pour les villes

- Évacuation des habitants
- Dégâts matériels
- Pollution de l'air
- Risque accru par le changement climatique

Introduction

Les feux de forêt : un danger pour les villes

- Évacuation des habitants
- Dégâts matériels
- Pollution de l'air
- Risque accru par le changement climatique

-> Simuler le phénomène pour mieux le comprendre : cas du feu de La Teste-De-Buche

Introduction

Modélisation par automate cellulaire

Défintion

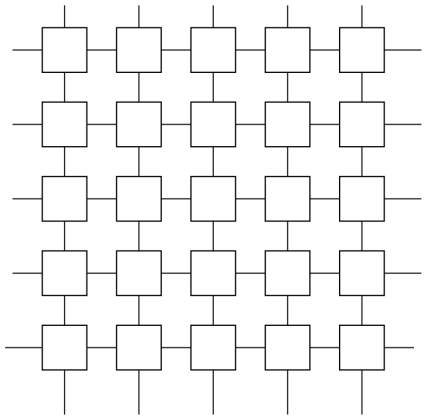
Un automate cellulaire consiste en un réseau de cellules contenant chacune un état qui peut évoluer au cours du temps. L'état d'une cellule au temps t+1 est fonction de l'état au temps t d'un nombre fini de cellules appelé son « voisinage ».

Introduction

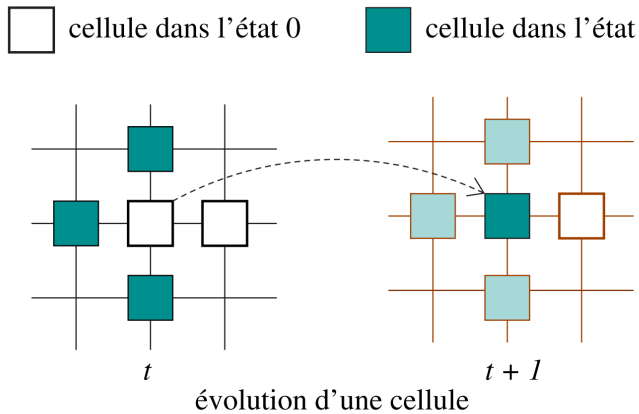
Modélisation par automate cellulaire

Défintion

Un automate cellulaire consiste en un réseau de cellules contenant chacune un état qui peut évoluer au cours du temps. L'état d'une cellule au temps t+1 est fonction de l'état au temps t d'un nombre fini de cellules appelé son « voisinage ».



un réseau de cellule



évolution d'une cellule

Présentation du modèle

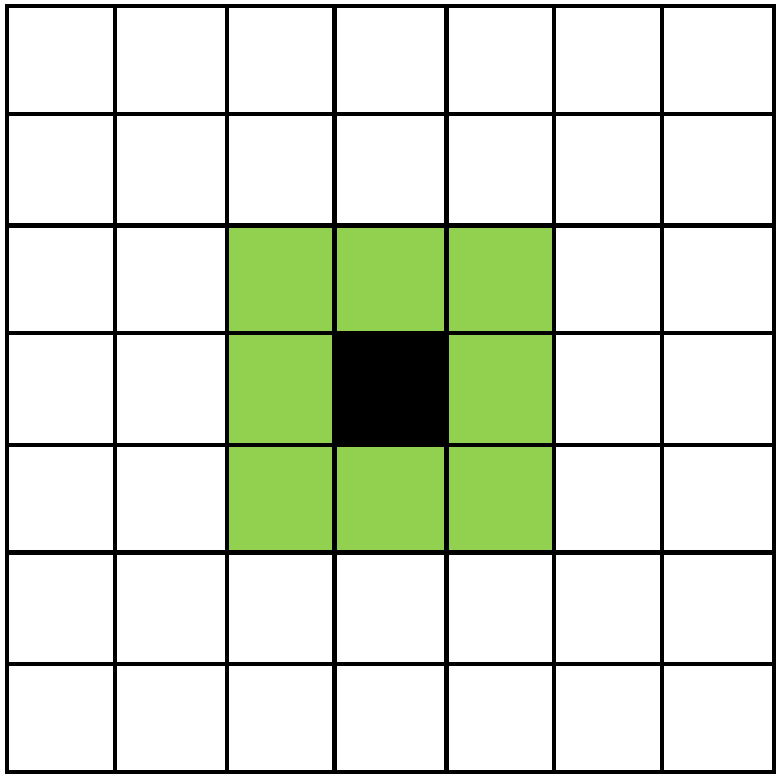
Règle de propagation

Toute case en feu peu ignifier une case voisine

Présentation du modèle

Règle de propagation

Toute case en feu peu ignifier une case voisine



Voisinage de Moore d'ordre 1

Les paramètres ...

des cellules

- Occupation du sol : Vide, cultivé, forêt, prairie

Les paramètres ...

des cellules

- Occupation du sol : Vide, cultivé, forêt, prairie
- Densité de végétation : Aucune, éparse, dense, très dense
- État : Ne brûle pas, brûle

Les paramètres ...

des cellules

- Occupation du sol : Vide, cultivé, forêt, prairie
- Densité de végétation : Aucune, éparse, dense, très dense
- État : Ne brûle pas, brûle

Implémentation: matrice d'entiers codés sur 8 bits

Les paramètres ...

des cellules

- Occupation du sol : Vide, cultivé, forêt, prairie
- Densité de végétation : Aucune, éparse, dense, très dense
- État : Ne brûle pas, brûle

Implémentation: matrice d'entiers codés sur 8 bits
Valeur = Occupation du sol + 4* Densité + 16 * État

Les paramètres ...

des cellules

- Occupation du sol : Vide, cultivé, forêt, prairie
- Densité de végétation : Aucune, éparse, dense, très dense
- État : Ne brûle pas, brûle

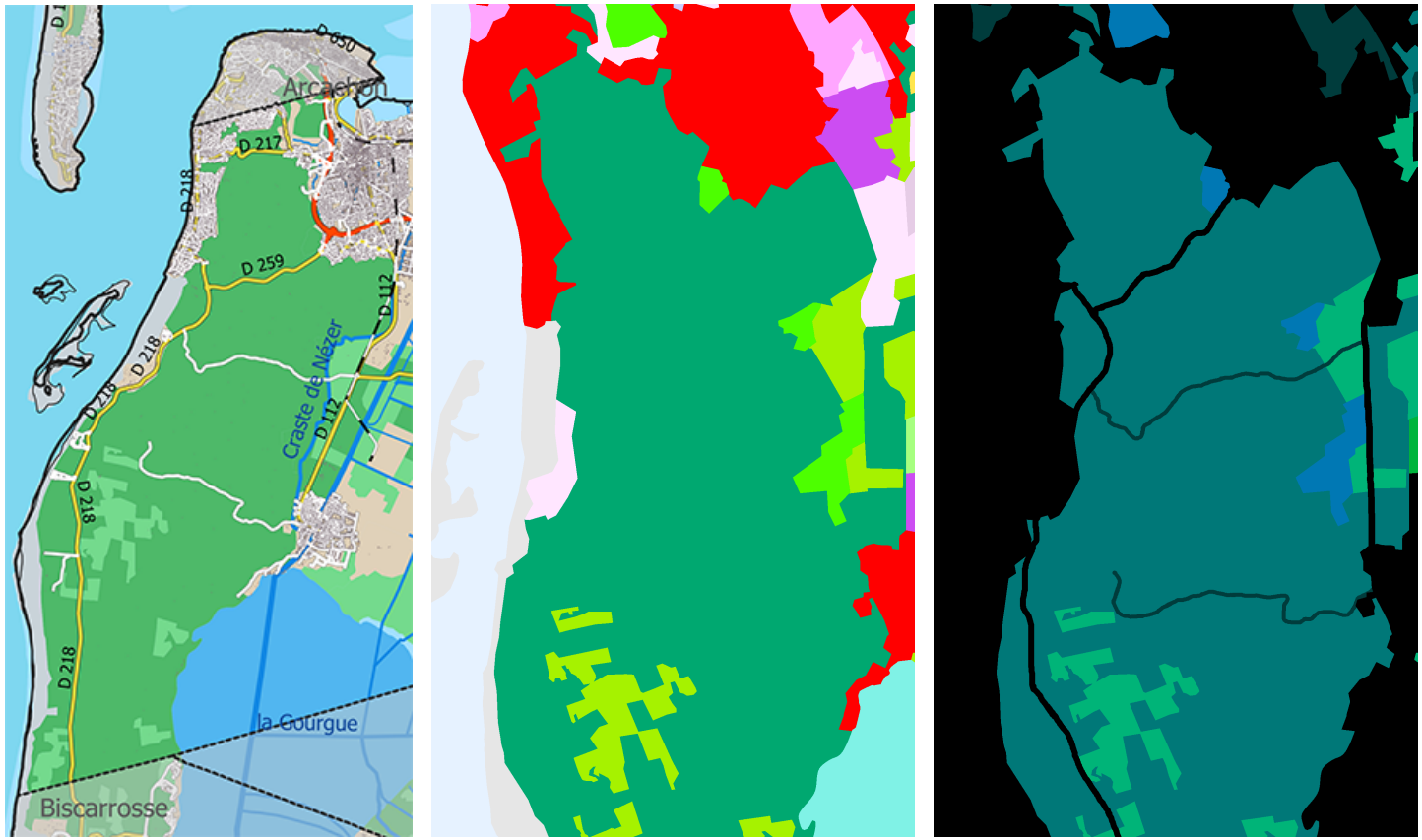
Implémentation: matrice d'entiers codés sur 8 bits
Valeur = Occupation du sol + 4* Densité + 16 * État

```
[[ 9. 29.  6.  7.]
 [26.  5.  8. 13.]
 [15. 13. 30. 14.]
 [24.  9.  5. 10.]]
```

Exemple de matrice

Les paramètres

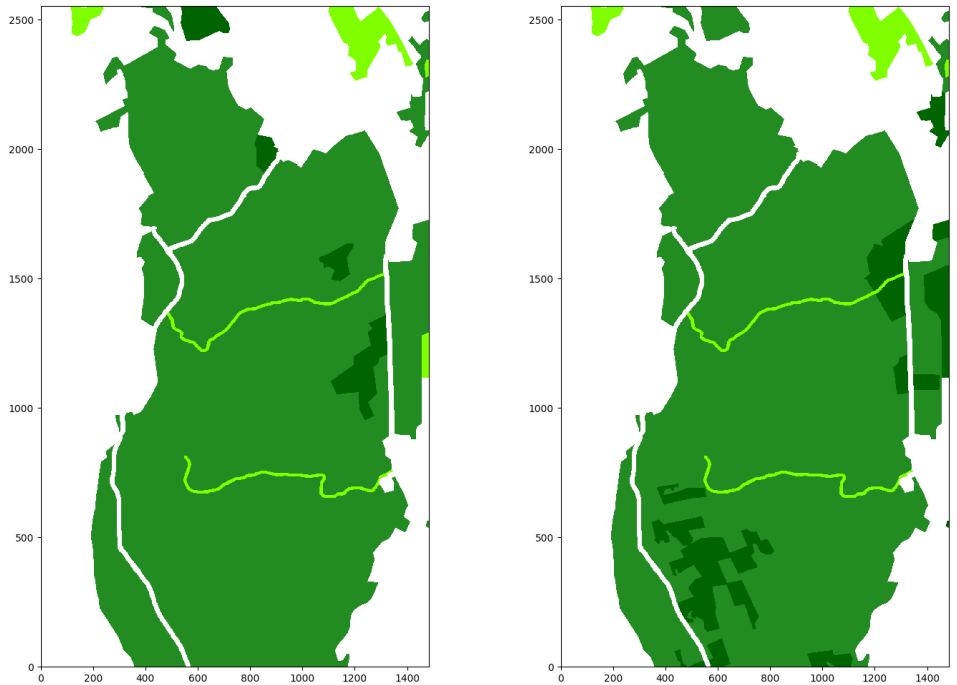
Récupération des données



De gauche à droite : OpenStreetMap, Corrine Land Cover, Carte modifiée

Les paramètres

Carte obtenue



Occupation des sols (à gauche), densité (à droite)

Les paramètres

de propagation

- Constante de probabilité p_h
- Coefficients liés aux type de cellule p_{veg} et p_{den}

Les paramètres

de propagation

- Constante de probabilité p_h
- Coefficients liés aux type de cellule p_{veg} et p_{den}

Vide	-1	*Aucune*	-1
Cultivé	-0.3	*Eparse*	-0.4
Forêt	0	*Dense*	0
Prairie	0.4	*Très dense*	0.3

Coefficients associés (p_{veg} et p_{den})

Les paramètres
de propagation

- Constante de probabilité p_h
- Coefficients liés aux type de cellule p_{veg} et p_{den}

$Vide$	-1	$Aucune$	-1
$Cultivé$	-0.3	$Eparse$	-0.4
$Forêt$	0	$Dense$	0
$Prairie$	0.4	$Très\ dense$	0.3

Coefficients associés (p_{veg} et p_{den})

- Impact du vent sur la propagation : p_w
- $p_{burn} = p_h p_w (1 + p_{veg})(1 + p_{den})$

Probabilité pour une case en feu d'ignifier un voisin :

$$\min(1,\ p_{burn})$$

Les paramètres

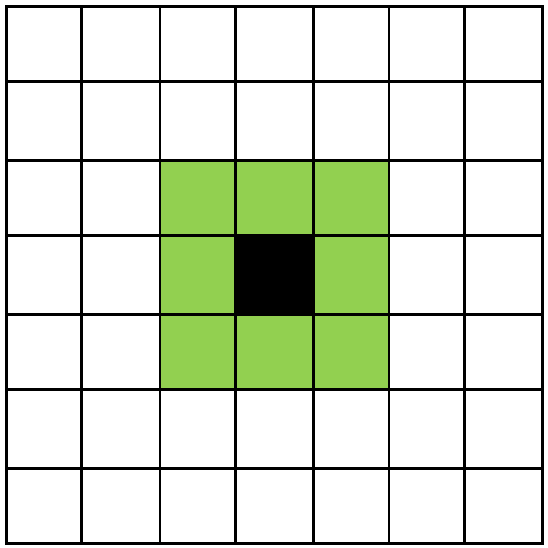
Le vent

- Trouver l'angle θ entre la direction du vent et celle du feu

Les paramètres

Le vent

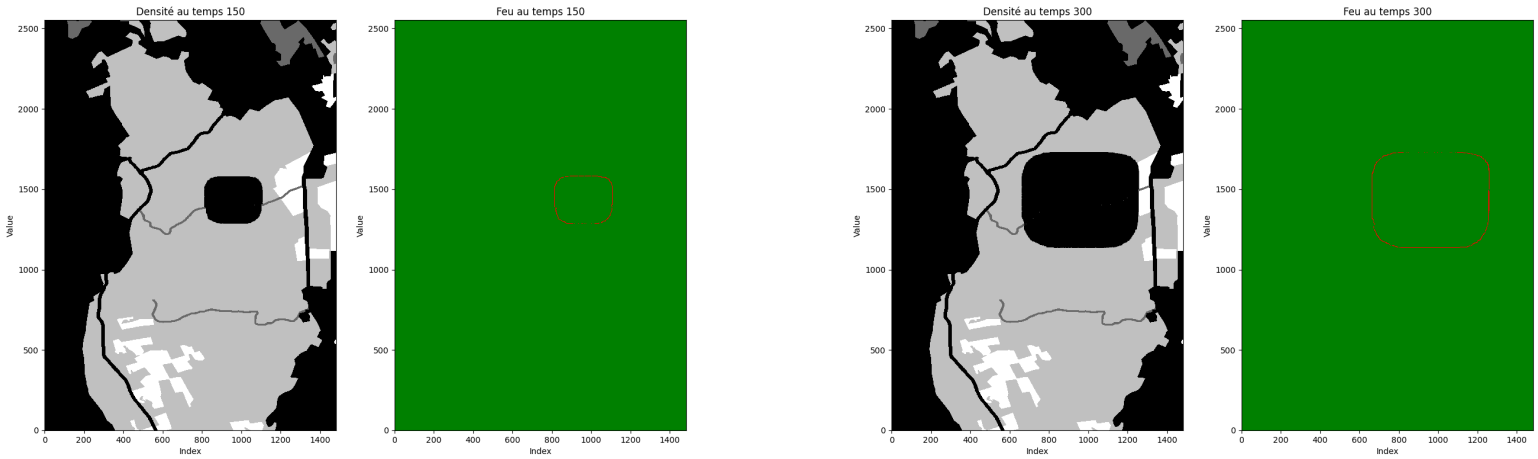
- Trouver l'angle θ entre la direction du vent et celle du feu
- Calculer $p_w = e^{V*(c1+c2(cos\theta\ -1))}$



Voisinage de Moore d'ordre 1

Simulation

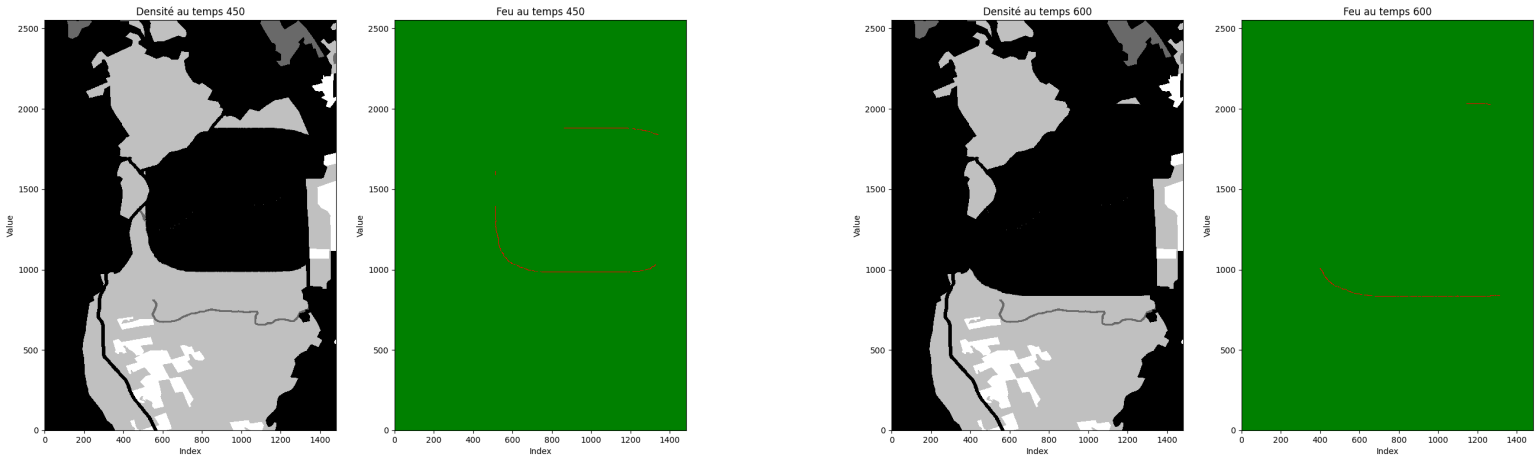
Sans vent



Feu du 13 au 14 juillet

Simulation

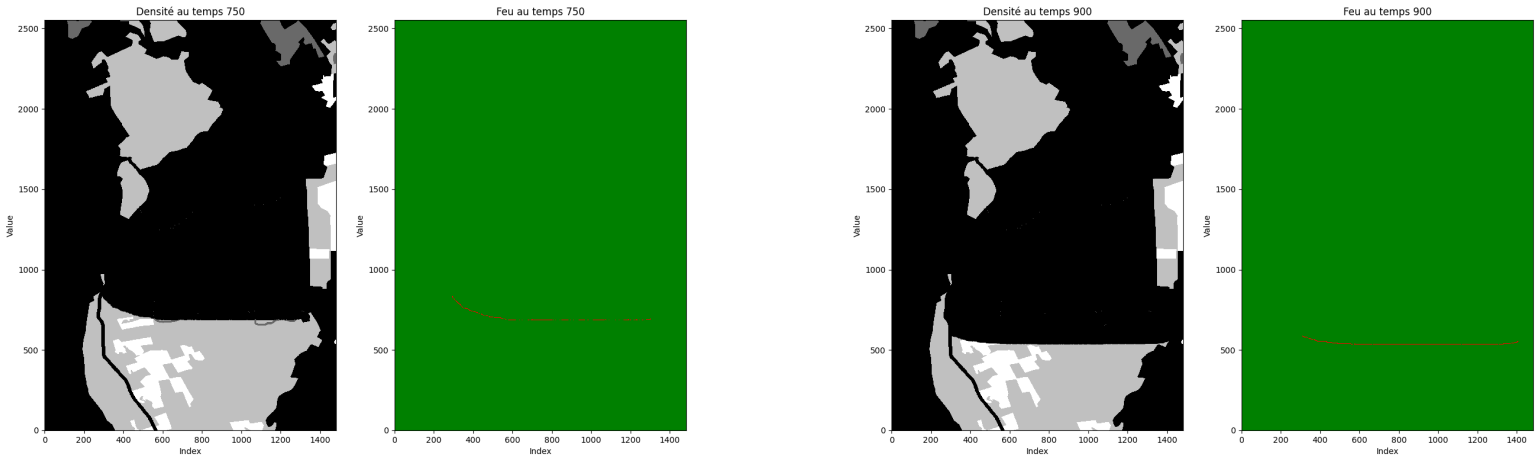
Sans vent



Feu du 15 au 16 juillet

Simulation

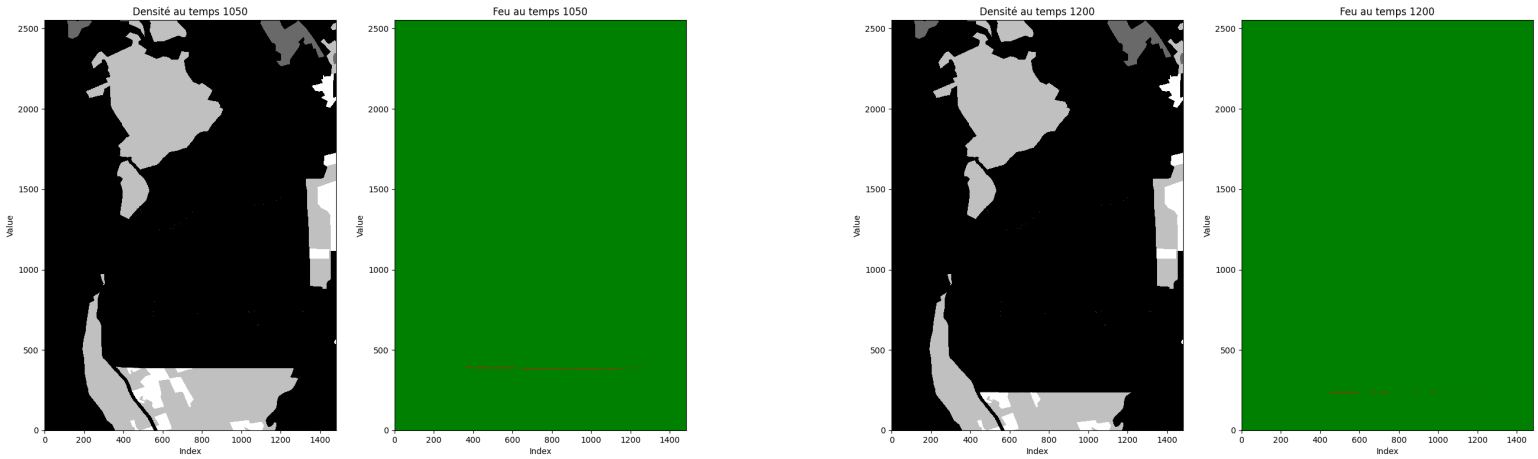
Sans vent



Feu du 17 au 18 juillet

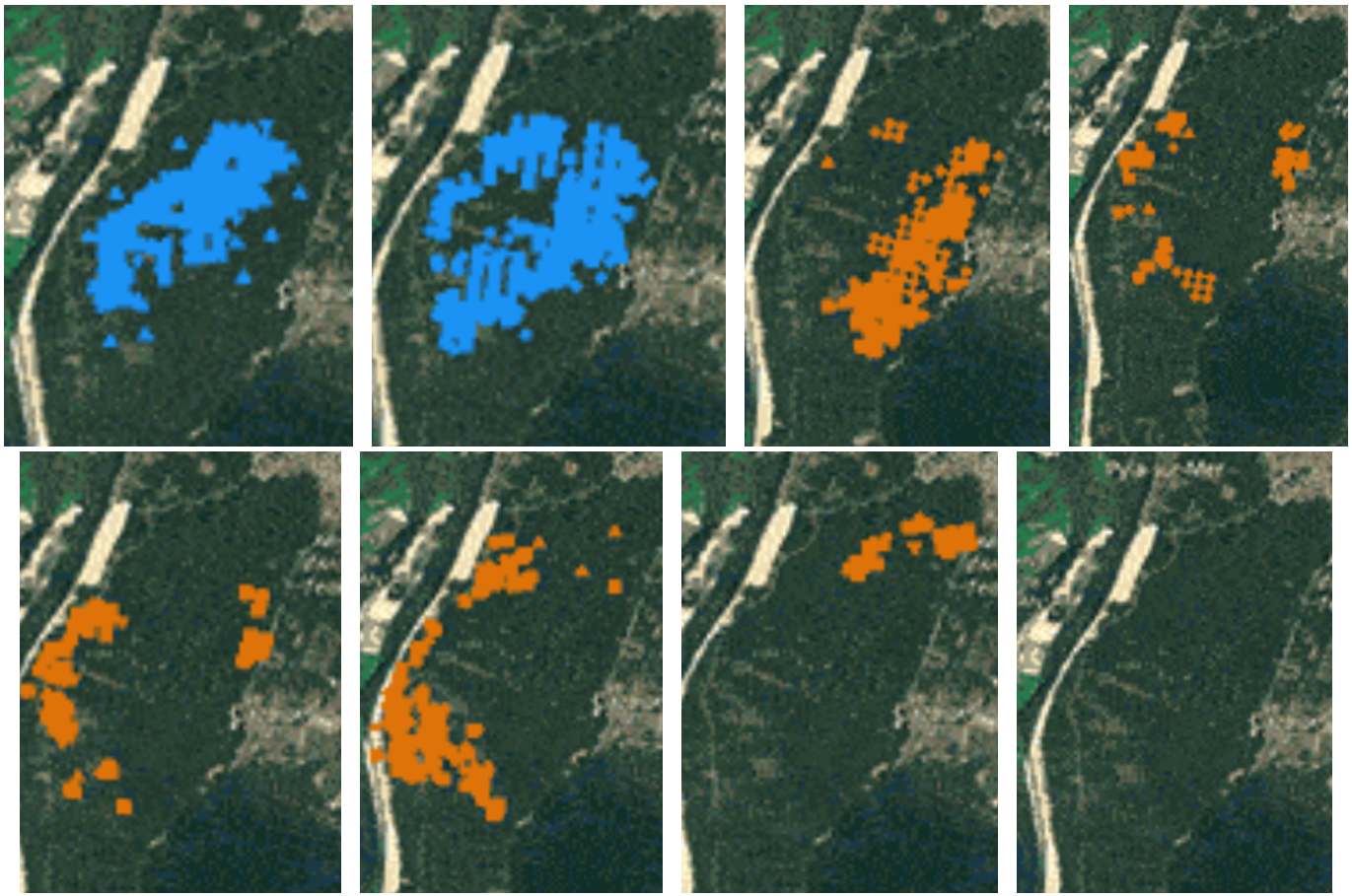
Simulation

Sans vent



Feu du 19 au 20 juillet

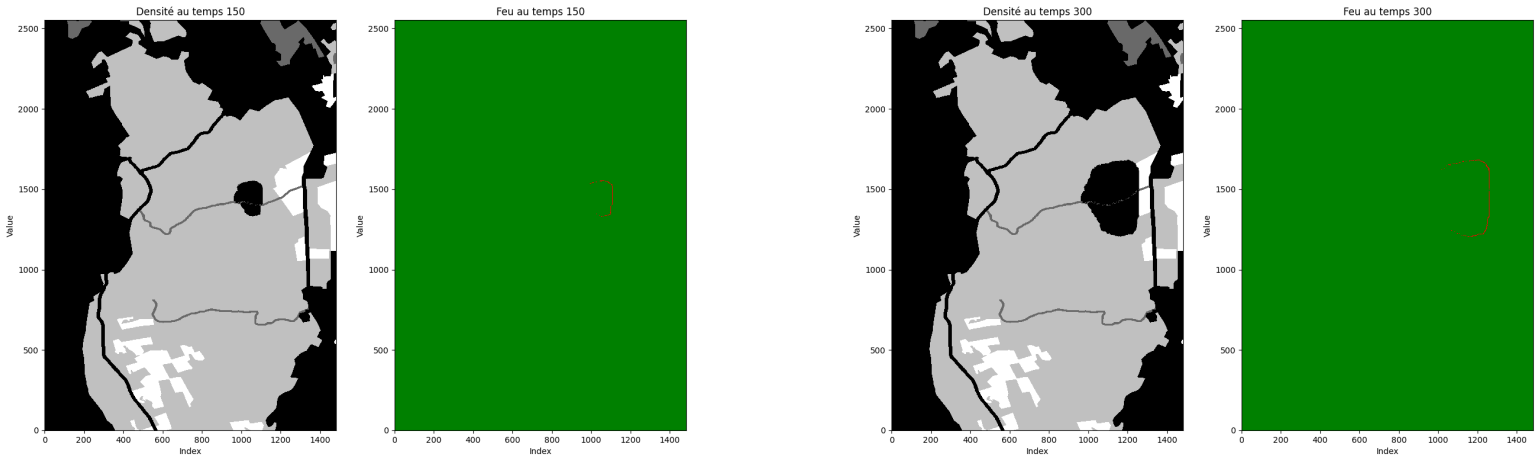
Comparaison avec le feu réel



Feu du 13 au 20 juillet (Images Forestopic)

Simulation

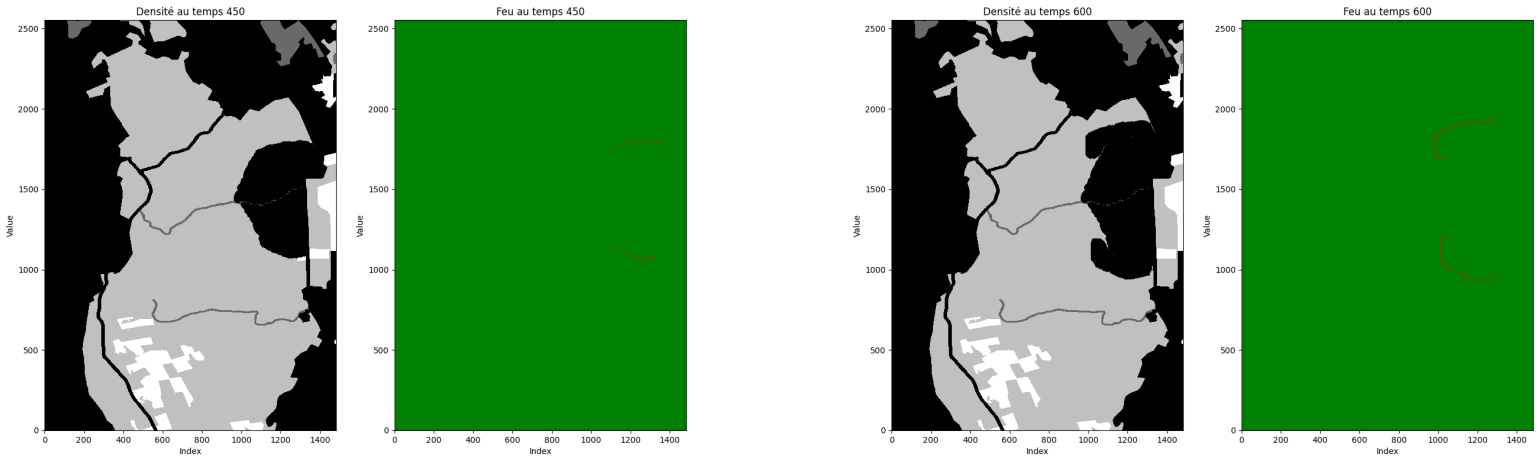
Avec vent



Feu du 13 au 14 juillet

Simulation

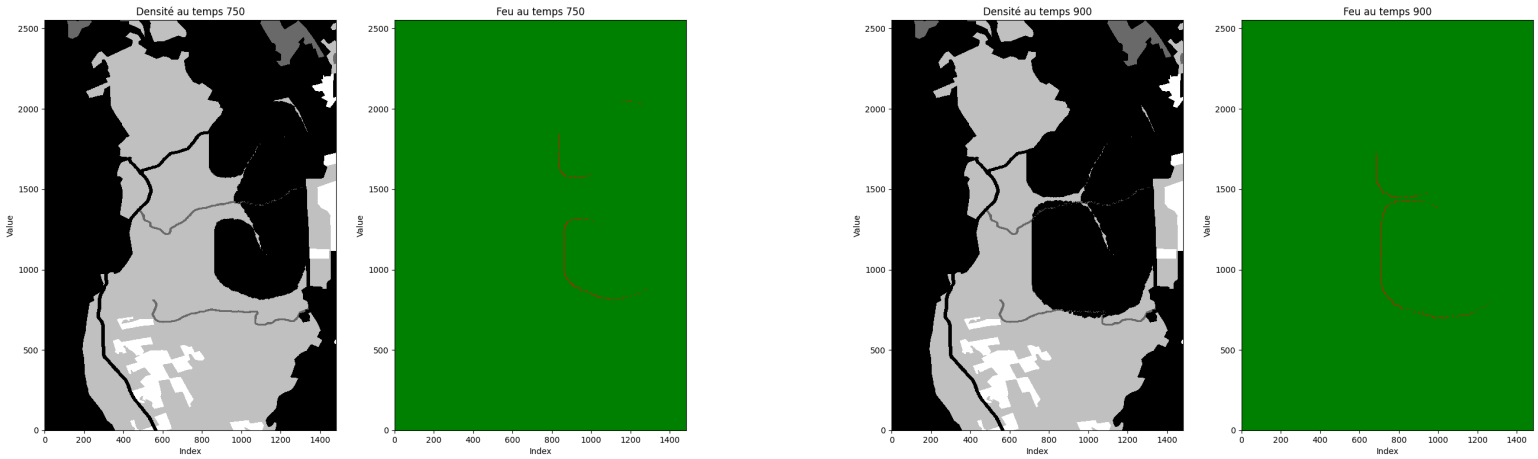
Avec vent



Feu du 15 au 16 juillet

Simulation

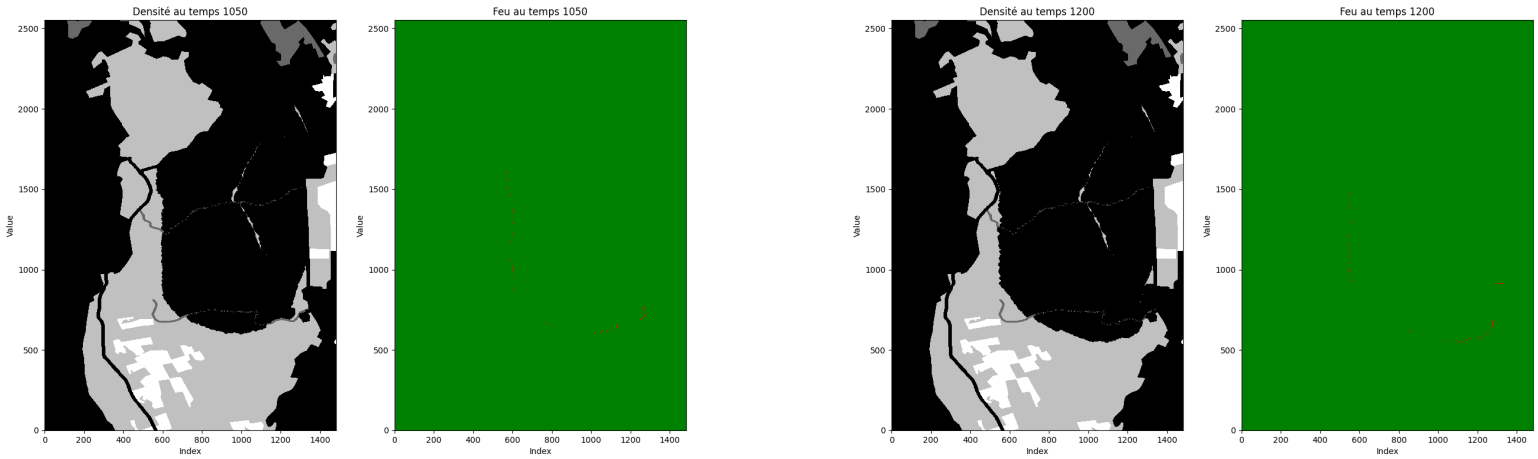
Avec vent



Feu du 17 au 18 juillet

Simulation

Avec vent



Feu du 19 au 20 juillet

Conclusion

Pistes d'amélioration

- Prise en compte du relief du terrain

Conclusion

Pistes d'amélioration

- Prise en compte du relief du terrain
- Prise en compte des efforts des soldats du feu

Conclusion

Pistes d'amélioration

- Prise en compte du relief du terrain
- Prise en compte des efforts des soldats du feu
- Prise en compte du phénomène de projection

Conclusion

Pistes d'amélioration

- Prise en compte du relief du terrain
- Prise en compte des efforts des soldats du feu
- Prise en compte du phénomène de projection
- Prise en compte de l'humidité

Conclusion

Pistes d'amélioration

- Prise en compte du relief du terrain
- Prise en compte des efforts des soldats du feu
- Prise en compte du phénomène de projection
- Prise en compte de l'humidité
- Préciser les paramètres

Conclusion

Pistes d'amélioration

- Prise en compte du relief du terrain
- Prise en compte des efforts des soldats du feu
- Prise en compte du phénomène de projection
- Prise en compte de l'humidité
- Préciser les paramètres
- Ajouter un paramètre qui décrit combien de temps brûle une case

Transformer une image en matrice

Python (Notebook)

```
import numpy as np
from PIL import Image as PIL_Image

# choisir un fichier image
fichierImage =  "Carte_coloriee_r.png"
#ouverture dans PIL
img = PIL_Image.open(fichierImage)
# Recuperation de la x_taille et y_taille de l'image
x_taille, y_taille = img.size
print(x_taille, y_taille)

def pixel2couleur(x, y) : #regarde la couleur du pixel
    couleur = img.getpixel((x, y))
    return couleur
def couleur2int8(a): #renvoie l'entier correspondant
    r,g,b = a
    return g/60 + 4*b/60

#creation de la carte
matrice = np.zeros((x_taille, y_taille), dtype=np.uint8)
for x in range(x_taille):
    for y in range(y_taille):
        matrice[x][y_taille-y-1] = couleur2int8(pixel2couleur(x, y))

#sauvegarde dans un fichier txt
out = open("out.txt", "w")
for x in range(x_taille):
        for y in range(y_taille):
            out.write(chr(matrice[x][y]+97))
out.close()
```

Propagation du feu I

Code C

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>

int x_taille = 1483; int y_taille = 2553; //taille de la carte
uint8_t rand_num; float rand_float;  //variables pour les fonctions random

#define PI 3.14159
int v = 10; float c1 = 0.045; float c2 = 0.191; float theta = -PI/2;
// vitesse du vent ; parametres de vent 1 et 2 ; direction du vent (rad)
int indice(int x,int y){return x*y_taille+y;}
//fonction qui renvoie l'indice d'une case de coordonnees (x,y) dans le tableau

float compute_veg(int val){
    int veg_type = (val >> 2) & 3; float p_veg;
    if (veg_type == 0) {p_veg = -1.;}
    if (veg_type == 1) {p_veg = -0.3;}
    if (veg_type == 2) {p_veg = .0;}
    if (veg_type == 3) {p_veg = 0.4;}
    return p_veg;}
// fonction qui renvoie la probabilite de bruler a cause de la vegetation
float compute_den(int val){
    int den_type = val & 3; float p_den;
    if (den_type == 0) {p_den = -1.;}
    if (den_type == 1) {p_den = -0.4;}
    if (den_type == 2) {p_den = .0;}
    if (den_type == 3) {p_den = 0.3;}
    return p_den;}
// fonction qui renvoie la probabilite de bruler a cause de la densite

float fire_angle(int x, int y){
    if (x==0 && y==1)  {return fmod(theta, 2 * PI);}
```

Propagation du feu II

Code C

```
    if (x==-1 && y==1) {return fmod(theta - PI/4, 2 * PI);}
    if (x==-1 && y==0) {return fmod(theta - PI/2, 2 * PI);}
    if (x==-1 && y==-1){return fmod(theta - 3*PI/4, 2 * PI);}
    if (x==0 && y==-1) {return fmod(theta - PI, 2 * PI);}
    if (x==1 && y==-1) {return fmod(theta - 5*PI/4, 2 * PI);}
    if (x==1 && y==0)  {return fmod(theta - 3*PI/2, 2 * PI);}
    if (x==1 && y==1)  {return fmod(theta - 7*PI/4, 2 * PI);}
} // fonction qui renvoie l'angle du feu par rapport au vent

bool catch_fire(int val, float angle) {
    float ph = .58;                                    // parametre
    float p_veg = compute_veg(val); // probabilite de bruler a cause de la vegetation
    float p_den = compute_den(val); // ou de la densite
    float p_vent = exp(v*(c1+c2*(cos(angle)-1))); // probabilite de bruler a cause du vent
    float p_burn = ph*p_vent*(1+p_veg)*(1+p_den); // probabilite de bruler
    rand_float = (float) rand() / RAND_MAX;        // random float entre 0 et 1
    return rand_float<p_burn;}                     // renvoie true si le feu prend

// Operateurs bit a bit : << n : decalage a gauche de n bits; & : et
void prop(uint8_t carte[]){
    bool *li = malloc(x_taille*y_taille*sizeof(bool));
    for (int i =0; i<x_taille; i++){
        for (int j=0; j<y_taille; j++){li[indice(i, j)] = 0;}
    } // tableau indiquant les cases qui vont bruler

for (int x =1; x<x_taille-1; x++){
    for (int y=1; y<y_taille-1; y++){
        if (((carte[indice(x, y)] >> 4) & 1 )==1){
        int val1 = carte[indice(x-1, y)]; int val2 = carte[indice(x, y-1)];
        int val3 = carte[indice(x+1, y)]; int val4 = carte[indice(x, y+1)];
        int val5 = carte[indice(x-1, y-1)]; int val6 = carte[indice(x+1, y-1)];
        int val7 = carte[indice(x+1, y+1)]; int val8 = carte[indice(x-1, y+1)];
```

Propagation du feu III

Code C

```
        if (!((val1 >> 4) & 1)){if (catch_fire(val1, fire_angle(-1, 0))) {li[indice(x-1, y)]=1;}}
        if (!((val2 >> 4) & 1)){if (catch_fire(val2, fire_angle(0, -1))) {li[indice(x, y-1)]=1;}}
        if (!((val3 >> 4) & 1)){if (catch_fire(val3, fire_angle(1, 0)))  {li[indice(x+1, y)]=1;}}
        if (!((val4 >> 4) & 1)){if (catch_fire(val4, fire_angle(0, 1)))  {li[indice(x, y+1)]=1;}}
        if (!((val5 >> 4) & 1)){if (catch_fire(val5, fire_angle(-1, -1))) {li[indice(x-1, y-1)]=1;}}
        if (!((val6 >> 4) & 1)){if (catch_fire(val6, fire_angle(+1, -1))) {li[indice(x+1, y-1)]=1;}}
        if (!((val7 >> 4) & 1)){if (catch_fire(val7, fire_angle(+1, +1))) {li[indice(x+1, y+1)]=1;}}
        if (!((val8 >> 4) & 1)){if (catch_fire(val8, fire_angle(-1, +1))) {li[indice(x-1, y+1)]=1;}}
    }}} // on parcourt les cases en feu et on regarde si les cases voisines vont bruler
    for (int x =0; x<x_taille; x++){
        for (int y=0; y<y_taille; y++){
            int val = carte[indice(x, y)];
            if (((val >> 4) & 1) ==1){
                    if ((val & 3) >0){carte[indice(x, y)] -= 1;}
                else {carte[indice(x, y)] -= 16 ;}
            }
            if (li[indice(x, y)]) {carte[indice(x, y)] += 16;
            }}} // on met a jour la carte
    free(li);  // on libere la memoire
}
void to_fichier(uint8_t (*liste)[x_taille][y_taille],int longueur ){
  FILE* fic = fopen("c_array.txt","w");
  for (int l = 0; l<longueur; l++){
    for (int i =0; i<x_taille; i++){
      for (int j = 0; j<y_taille; j++){
            fprintf(fic, "%c", liste[l][i][j]);
      }}}
    fclose(fic);
}  // on ecrit la liste dans un fichier
void tabcpy(uint8_t carte[], uint8_t (*liste)[x_taille][y_taille], int where){
```

Propagation du feu IV

Code C

```
  for (int i = 0; i < x_taille; i++){
    for (int j = 0; j< y_taille; j++){
      liste[where][i][j] = carte[indice(i, j)];
    }
  }
}  // on copie la carte dans la liste

int main()
{
    uint8_t *c = malloc(x_taille*y_taille*sizeof(uint8_t));
    FILE *file = fopen("out.txt", "r");
    uint8_t matrix[x_taille][y_taille];
    for (int x = 0; x < x_taille; x++) {
        for (int y = 0; y < y_taille; y++) {
            matrix[x][y] = (int) fgetc(file);
            matrix[x][y] -= 97;
            c[indice(x, y)] = matrix[x][y];
        }}
    fclose(file);
    // on lit la carte obtenue par python et on la met dans un tableau
    c[indice(960, 1433)] +=16;   // position estimee du depart de feu

    uint8_t (*liste)[x_taille][y_taille] = malloc(2000 * sizeof(*liste));
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < x_taille; j++) {
           for (int k = 0; k < y_taille; k++) {
               liste[i][j][k] = 0;
    }}} // initialisation de la liste de sauvegarde des cartes

    tabcpy(c, liste,0);
    for (int i = 1 ; i< 2000; i++) {
        prop(c); tabcpy(c, liste, i);
        if (i == 500){theta = +PI/2;}
```

Propagation du feu V

Code C

```
            if (i == 1000){theta = -PI/4;}
        }
        to_fichier(liste, 2000); // sauvegarde de la liste dans un fichier
        free(liste);
        free(c); // liberation de la memoire allouee
        return 0;
}
```

Affichage de l'automate cellulaire I

Python (Notebook)

```
import numpy as np
import random as rd

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

#Initialisation des couleurs de carte
cmap_veg = ListedColormap([ "white","chartreuse", "forestgreen","darkgreen"])
cmap_den = ListedColormap(["black", "dimgrey", "silver", "white"])
cmap_feu = ListedColormap(["green", "red"])

#ouverture du fichier txt (un peu long)
matrice = np.zeros((2000, 1483, 2553), dtype = np.uint8)
f = open("c_array.txt", "r")
for i in range(2000):
    for j in range(1483):
        for k in range(2553):
            matrice[i][j][k] = np.uint8(ord(f.read(1)))
f.close()

# Plot du type de vegetation (constant)
fig = plt.plot(squeeze = False, figsize = (7, 9))
plt.pcolormesh(matrice[0].T >> 2 & 3, cmap = cmap_veg)
plt.show()

# Plot en fonction du temps (densite et feu)

def plot(time_step):
    fig, ax = plt.subplots(1, 2, squeeze=False, figsize=(14, 9))

    ax[0][0].pcolormesh(matrice[time_step].T& 3, cmap = cmap_den)
    ax[0][0].set_xlabel('Index')
    ax[0][0].set_ylabel('Value')
    ax[0][0].set_title('Densitee␣au␣temps␣{}'.format(time_step))

    ax[0][1].pcolormesh((matrice[time_step].T >> 4) & 1, cmap = cmap_feu)
    ax[0][1].set_xlabel('Index')
```

Affichage de l'automate cellulaire II

Python (Notebook)

```
    ax[0][1].set_ylabel('Value')
    ax[0][1].set_title('Feu au temps {}'.format(time_step))
    plt.show()
# Slider pour la selection du temps
from ipywidgets import interact, IntSlider
interact(plot, time_step=IntSlider(min=0, max=len(matrice), step=1, value=0))
```