Gautier Kasperek
Master IVI

Rapport NIHM

TP : Encoder l'information avec des marques

Introduction

Dans ce TP nous allons étudier les fondamentaux de la visualisation d'information. Nous découpons ce rapport en 3 parties qui traiteront chacune des séances passés sur ce TP. Dans un premier temps nous aborderons la lecture et l'affichage de données simple, puis l'encodage d'information grâce aux marques et enfin les interactions avec le résultats choisis. Ce TP est aussi l'occasion de découvrir processing et donc d'expérimenter les possibilités en termes de représentation de données.

Séance 1 : Les des données et les afficher à l'écran

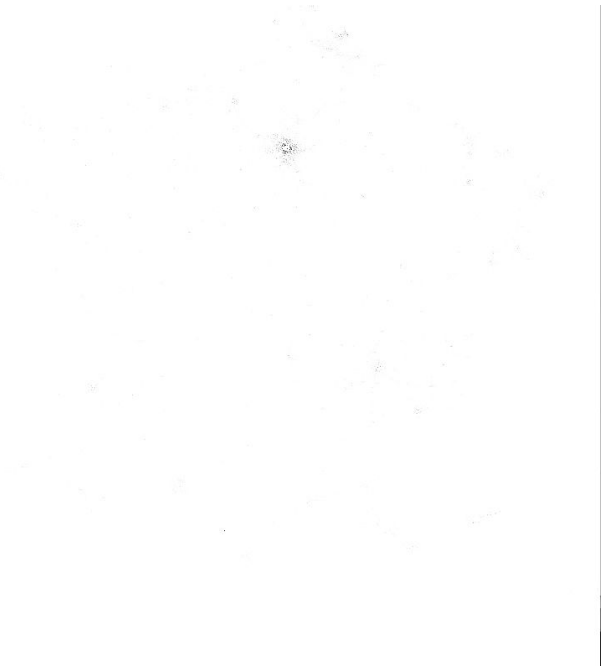
Les données utilisés lors de ce TP sont issus d'un fichier tsv représentant les villes en France. Chaque ville possède différentes informations tels que le code postal, le nom de la ville, ses coordonnées x et y, sa population, son altitude et sa surface. Les coordonnées de chaque ville sont transposés entre 0 et 800 afin de tenir dans la fenêtre d'affichage de l'application. Nous créons une instance de *City* pour chaque ligne parsé. Puis on affiche les villes dans la fenêtre.

Altitude Représentation de l'altitude

Nous avons ensuite représenté les villes en fonctions des altitudes. C'est-à-dire, nous avons mappés entre 0 et 255 chaque altitude en fonction de l'altitude minimum et maximum.

Densité

Nous avons ensuite réalisé la même manipulation en fonction de la densité de population. La densité de population étant calculé grâce à la surface et à la population de chaque ville.



Conclusion séance 1

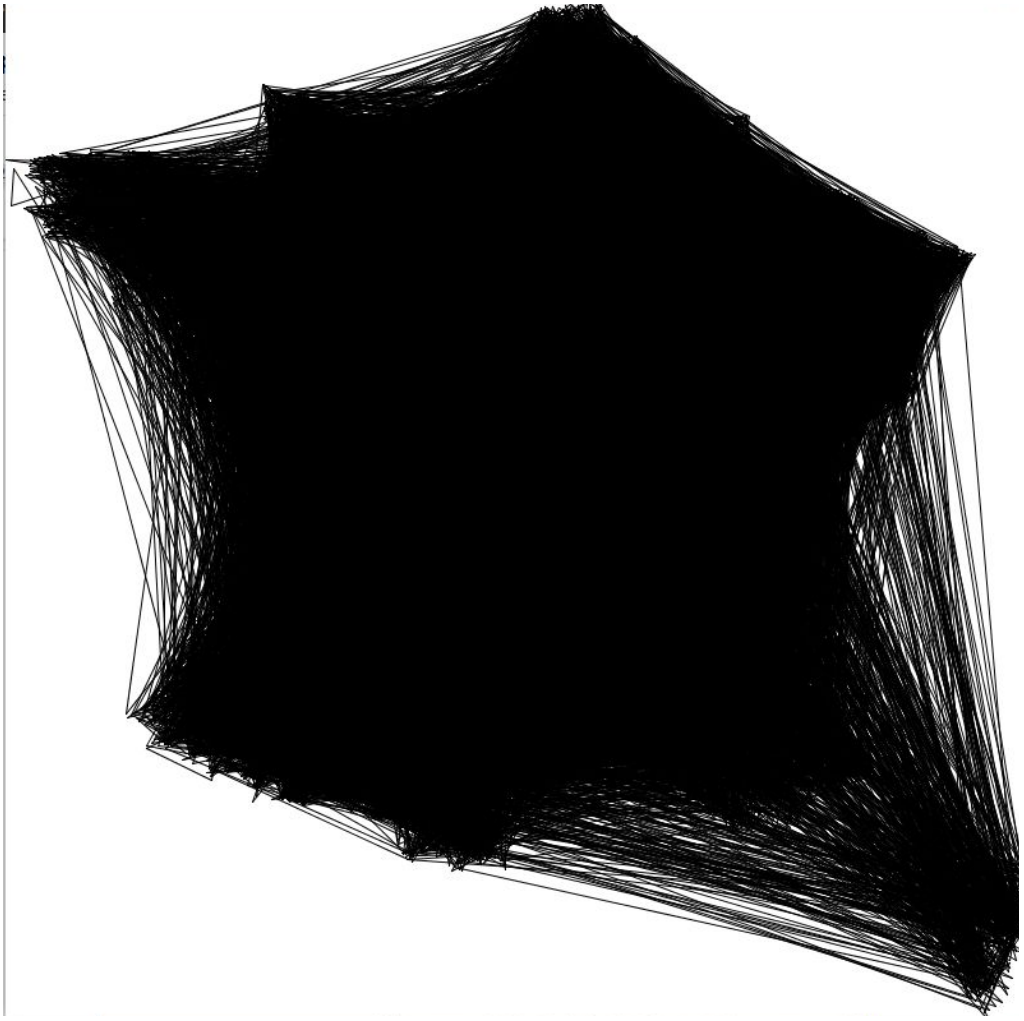
Comme nous pouvons le constater dans les deux exemples réalisés, la représentation est mauvaise et la plupart des villes sont à peine visible. Dans les deux cas c'est à cause de la fonction de mappage, les grandes altitudes, ou densité, sont trop élevés et écrasent les autres valeurs, d'où les représentations presque intégralement blanches. Nous pouvons ainsi constater l'importance des marquages dans la visualisation d'information, c'est pourquoi nous leur apporterons une attention particulière dans la partie 2.

Séance 2 : Encoder l'information avec des marques

Dans cette partie nous allons mettre en lumière les différentes visualisations que nous avons réalisé lors de la seconde séance.

Dans les ténèbres les lier

Afin de découvrir Processing et les possibilités de marqueurs nous avons souhaité relier chaque ville avec sa ville suivante dans l'ordre alphabétique.



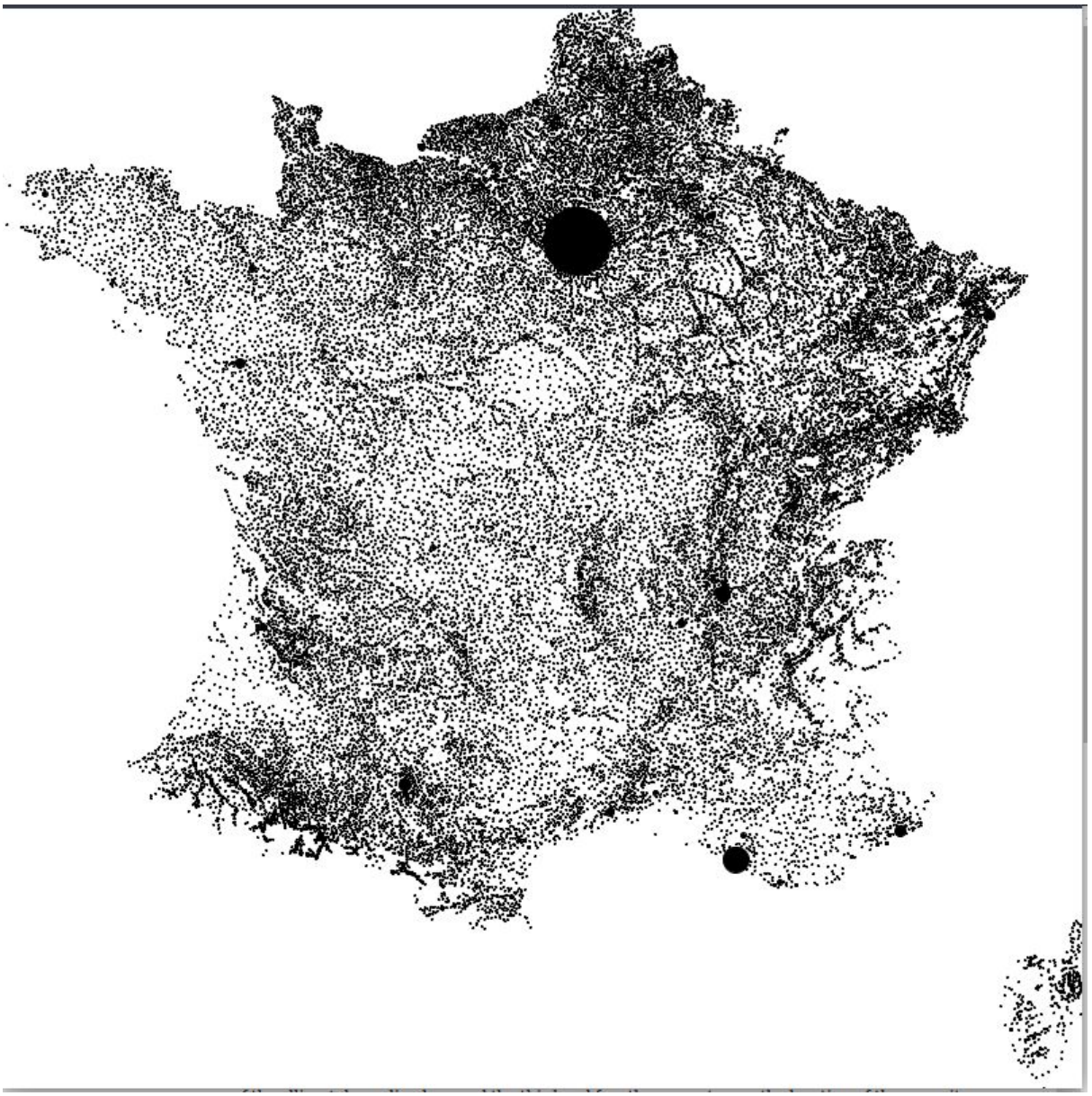
```
void draw(){
  background(255);
  //in your draw method
  //println(cities.length);
  for (int i = 0 ; i < cities.length -3 ; i++){
   // cities[i].draw();
     line(cities[i].x, cities[i].y, cities[i+1].x, cities[i+1].y);
  }
```

```
}
```

Cette représentation ne donne bien évidemment rien d'exploitable et est inutile si ce n'est que pour tester les possibilités de processing.

Taille d'ellipse en fonction de la population

Pour cette représentation nous revenons aux types du premiers essais. Cette fois nous représentons la population par la taille de l'ellipse représentant la ville.

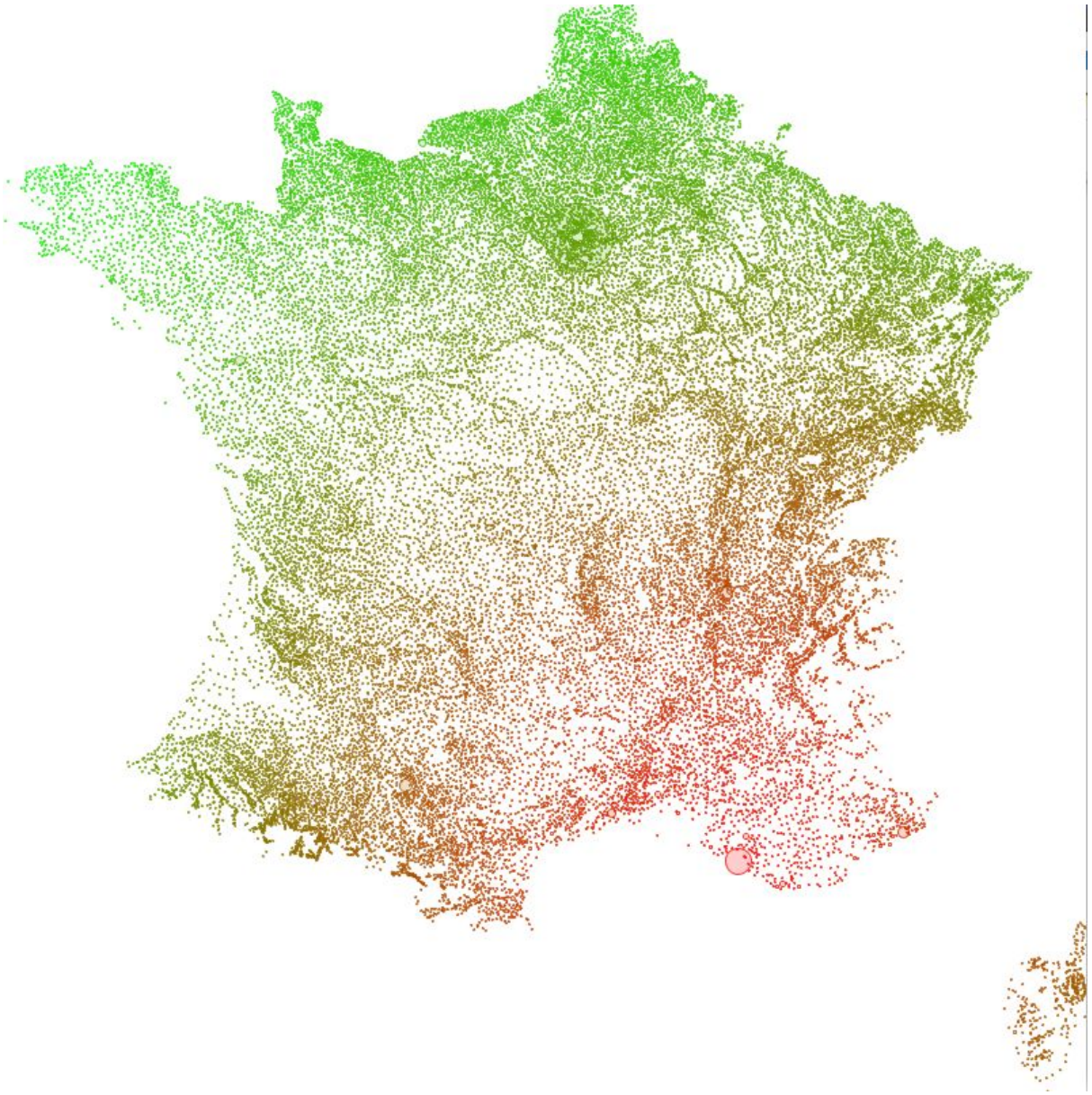


```
int populationToSize(){
        return int (map(int (population), minPopulation,maxPopulation,
                        1,50));
}
```

```
// City.draw()
 void draw(){
        fill(0);
        ellipse(x, y, populationToSize(), populationToSize());
         //set((int) x, (int) y, color(0));
      }
```

Représentation de la distance à vol d'oiseau

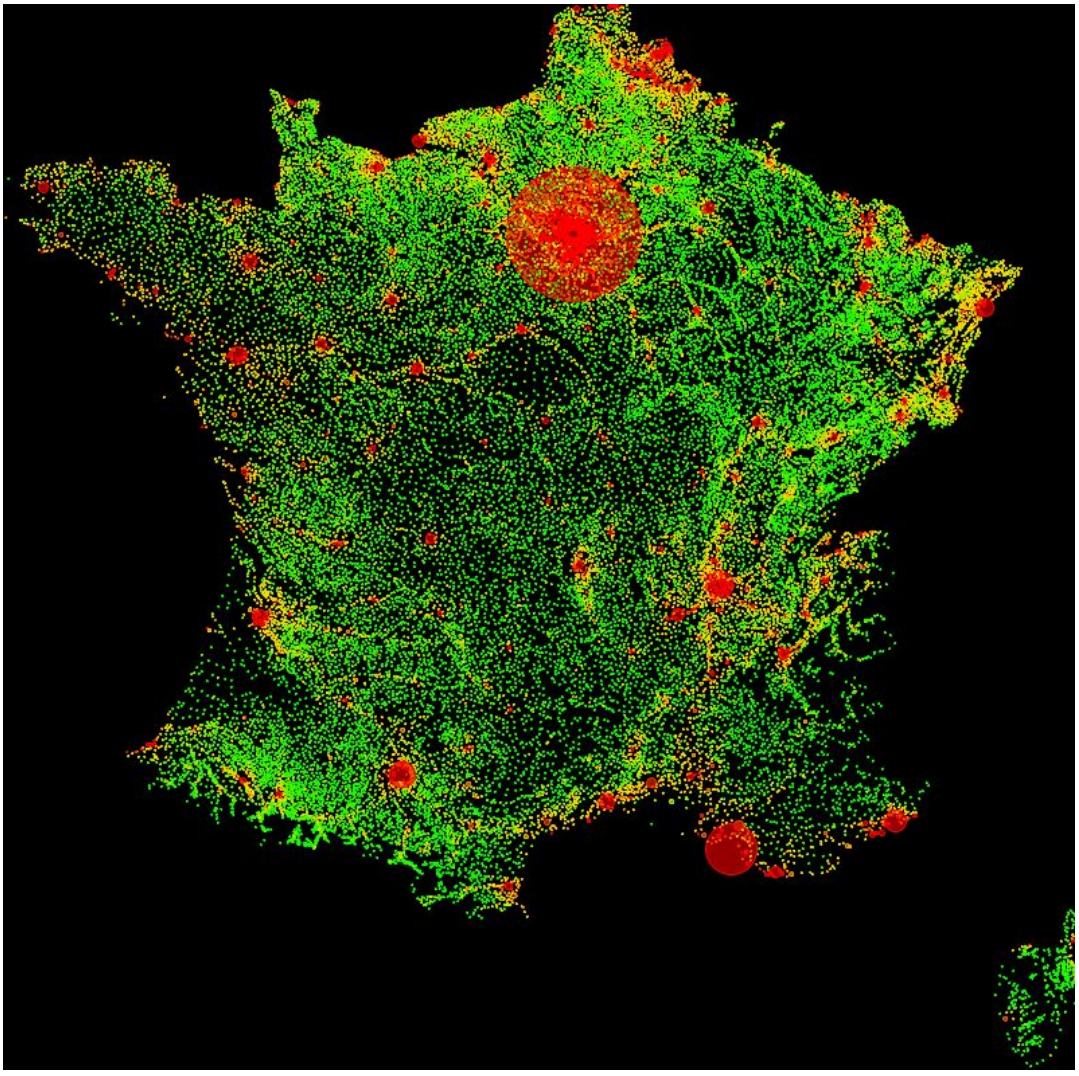
Puisque l'on connaît les coordonnées x et y dans un repère il est possible de calculer les distances à vol d'oiseau entre deux villes. nous choisissons de représenter la distance entre chaque ville et Marseille par un dégradé de vert (éloigné) et rouge (proche).



```
float distanceTo(City city){
   return sqrt(pow((city.x - this.x),2) + pow((city.y -
                                        this.y),2));
}

color distanceToChosenToColor(){
  return color(map(int (distanceTo(chosen)), 0, 750,
                255,0),map(int (distanceTo(chosen)),
                0, 750, 0,255),0);
}
```

Représentation de la densité de population et de la population de chaque ville

Dans cette représentation nous avons choisis deux marqueurs différents pour représenter deux données différentes. Dans un premier temps, la densité est représenté par le dégradé de couleur vert (faible), jaune et rouge (élevé).



```
color densityToColor(){
      color c = color(map(int(density), 0, 100,
                0,255),map(int(density), 0, 1000, 255,0),0);
      return color(c);
    }

    int populationToSize(){
      return int (map(int (population),minPopulation,maxPopulation,
                  1,100));
    }
```

En plus d'avoir utiliser un double dégradé de couleur nous avons fixé la densité maximum à 1000 afin que la valeurs de densité de Paris n'écrase une nouvelle fois les autres représentations. De plus, nous avons représenté les ellipse en transparent pour éviter que leur taille brouille certaines informations lorsqu'elles se trouvent proche d'une ville très peuplé. Nous avons utilisé cette carte afin de vérifier l'allure de la représentation obtenu. https://www.populationdata.net/cartes/france-densite-communes/

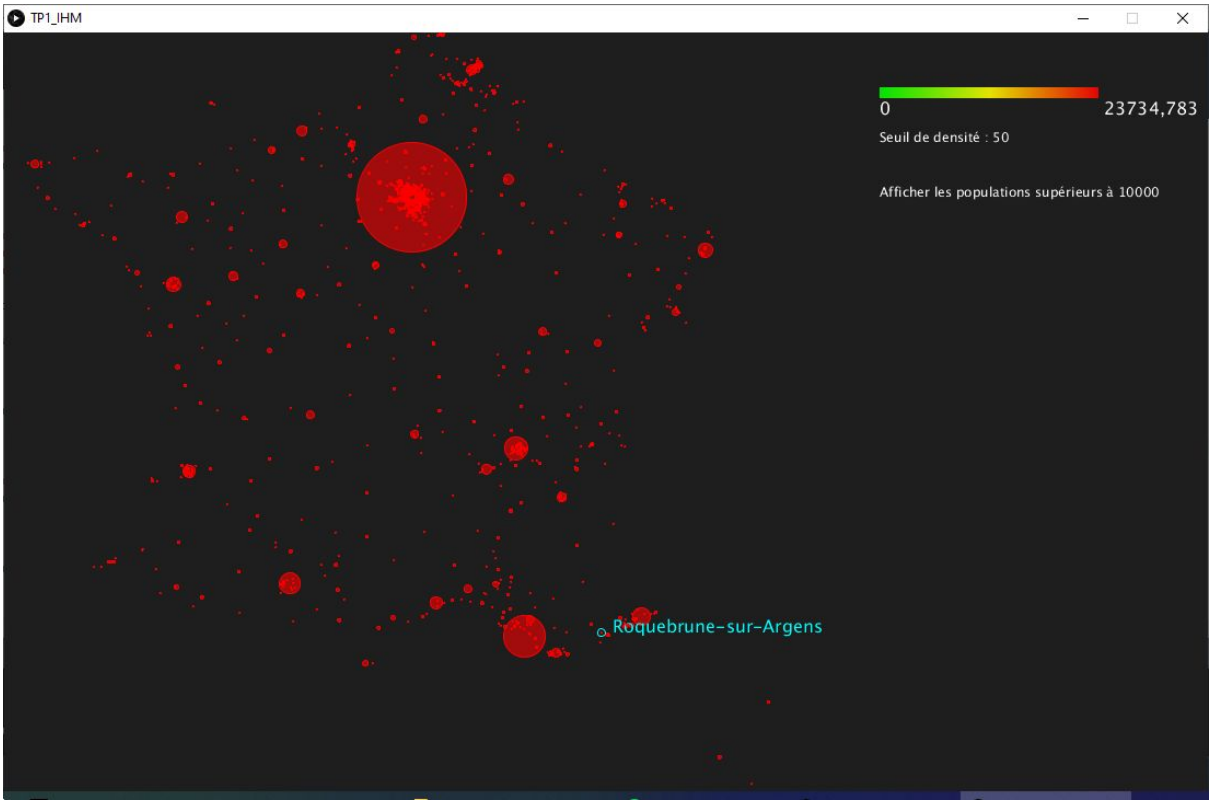
Séance 3 : Interactions

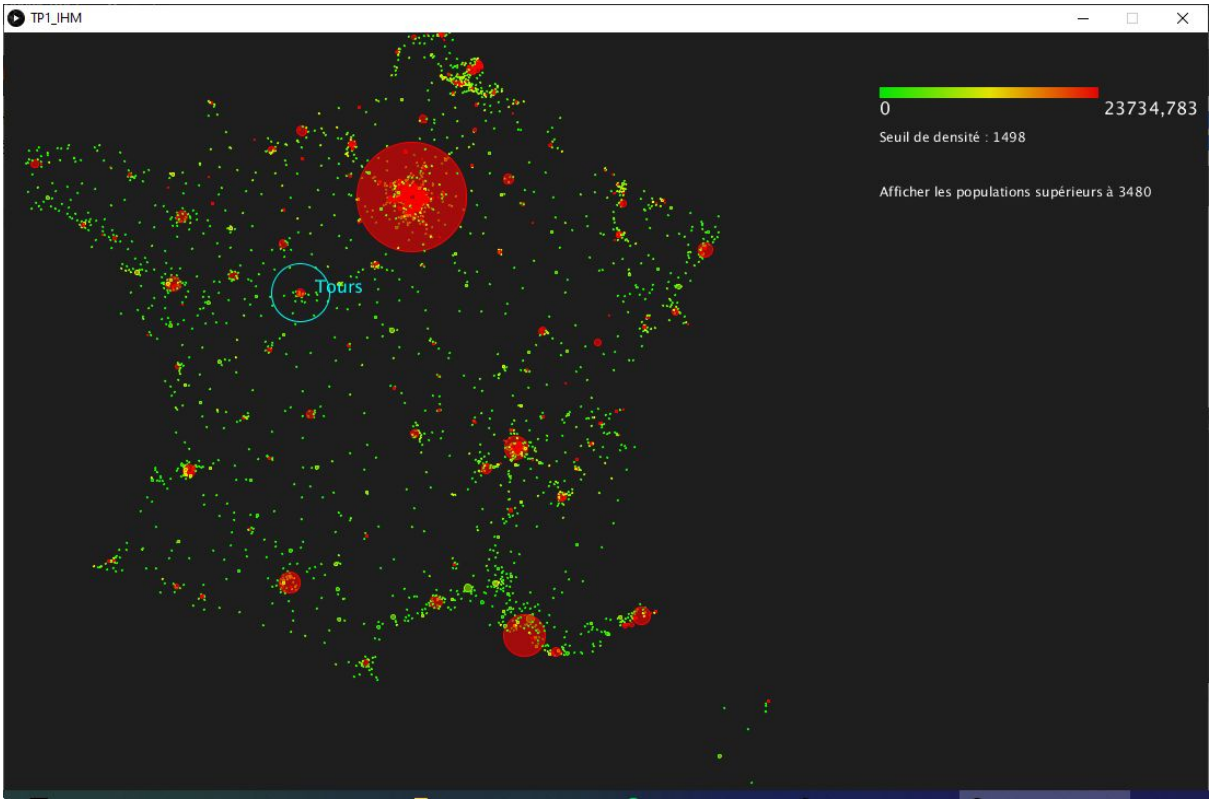
Pour réaliser les interactions nous avons utilisés la dernière représentation présenté. Cependant, pour utiliser la valeur de densité maximum, nous avons fixé un seuil pour sur lequel nous effectuerons les dégradés de couleur (vert -> jaune et jaune -> rouge). De plus, il est possible de modifier ce seuil afin de mettre en valeur les villes possédant une forte densité ou une faible densité. De plus il est maintenant possible de définir à partir de quel montant de population une ville est visible. Cela permet d'afficher par exemple que les plus grandes villes et/ou filtrer les petits villages qui peuvent montrer moins d'intérêt. De plus lorsque l'on passe la souris sur une ville, un cercle bleu ainsi que son nom s'affiche autour pour définir sa sélection.

```
void draw(){
        color c = densityBorneToColor();

// Affichage classique de la ville
        fill(c,150);
        stroke(c,500);
        ellipse(x, y, populationToSize(), populationToSize());
        if(isSelected){

// Animation autour de la ville selctionné
          if(selecteur > 50){
            selecteur = 0;
          }else{
            if(selecteur < 10){
              selecteur = selecteur +1;
            }
            else if(selecteur < 40){
              selecteur = selecteur + 4;
            }else{
              selecteur = selecteur +2;
            }
          }
          noFill();
          stroke(color(0,255,255),500);
          ellipse(x, y, populationToSize()+selecteur,
populationToSize()+selecteur);

// Affichage du nom de la ville
          fill(color(0,255,255));
          textAlign(LEFT);
          textSize(16);
          text(name, x + populationToSize()/2 + 10, y);
```

```
        }
    }
```

```
// Densité en couleur avec un seuil
      color densityBorneToColor(){
        color c = color(0);
        float mid = seuilDensity;
        float min = minPopulationToDisplay;

        if(this.density < mid){
          c = color( map(int(density), 0,mid, 0,255), 255,0);
        }else{
          c = color( 255,map(int(density), 0,mid, 255,0),0);
        }

        return c;
      }
```





Dans notre application, il est possible de :

- Réduire et augmenter le seuil de densité avec les touches du clavier Q et S.
- Réduire et augmenter le seuil d'habitant des villes à afficher avec A et Z
- Changer la couleur de fond avec Tab
- Afficher le nom d'une ville en passant sa souris dessus.