



# RAPPORT de Projet

Camille COUÉ , Victor COUR , Erwan KESSLER

*December 2018*

## Sommaire

- Introduction
- L'organisation pour répartir le travail
- Gestion des étapes
- Production finale
- Ce que le projet a apporté à chacun
- Remerciements
- Sources
- État de l'art

## 1 Introduction

Le but du projet est de créer un code qui permettrait de choisir la représentation (parmi celles qui sont proposées) où l'on placerait les différents aéroports du monde que le site openflight.org a ressuscité en 2017. On pourrait aussi choisir de centrer cette représentation sur une zone souhaitée.

Nous avons profité du lendemain de l'annonce du projet pour nous donner rendez-vous dans un restaurant afin de faire connaissance. Après avoir défini l'heure de notre première réunion pour fixer l'organisation du projet , Erwan nous a suggérer de créer un Trello (plateforme de gestion de projet) en attendant que la plateforme Git soit configurée.

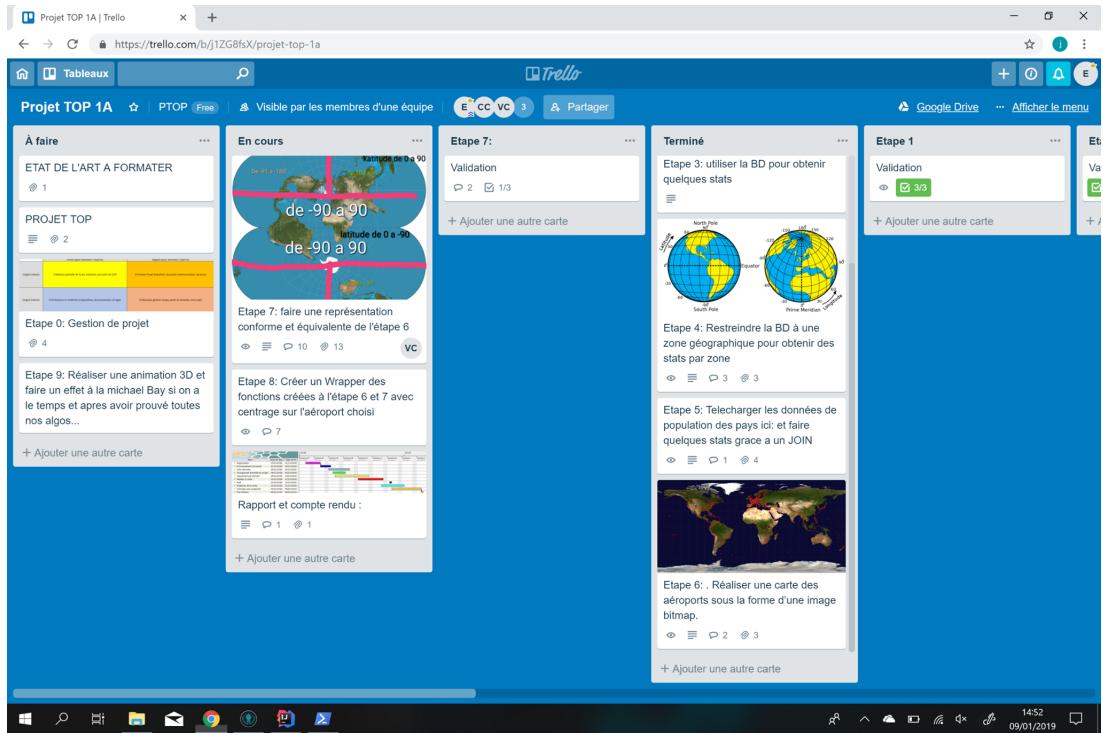


Figure 1: Trello du projet TOP 1A

## 2 L'organisation pour répartir le travail

Nous avons décidé pour hiérarchiser notre équipe de définir un chef de projet et nous avons convenu que Erwan remplirait ce poste. Ainsi, Camille et Victor joueront les rôles de secrétaires pour rédiger les comptes rendus à tour de rôle. (Listes des comptes rendus en annexe)

La première étape pour se répartir le travail a été de prévoir la durée des tâches, nous avons d'abord effectué un GANTT prévisionnel pour avoir une idée des étapes les plus coûteuse en temps, et à l'inverse, celles qui n'en demandaient pas énormément.

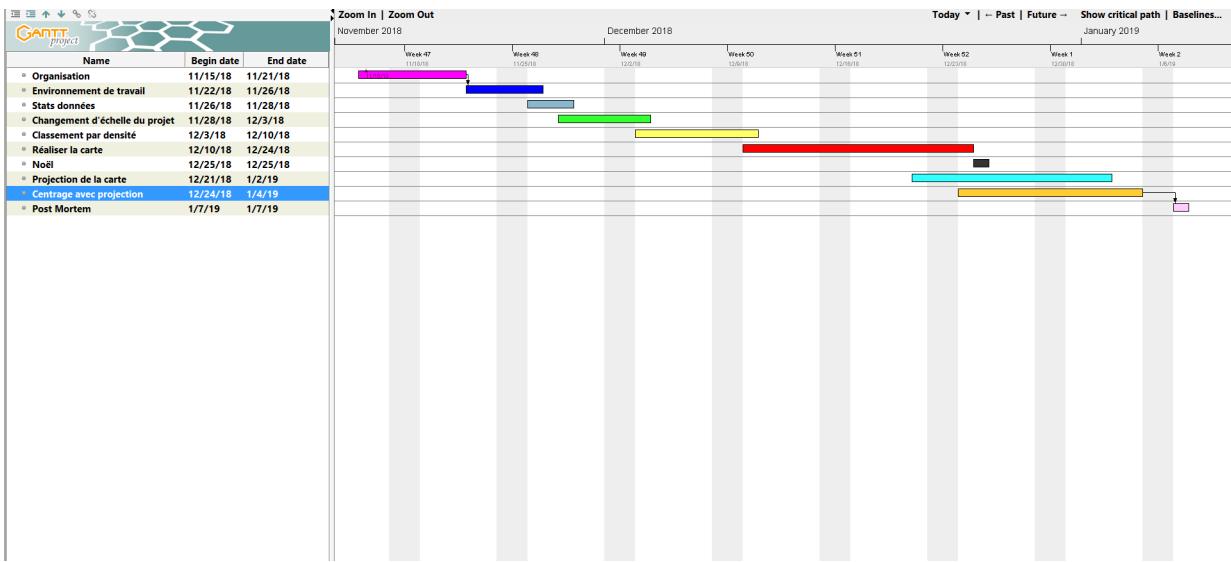


Figure 2: Gantt prévisionnel

Nous avons ensuite réparti les tâches avec un responsable pour chacune d'entre elles grâce à une matrice RACI.

Tâches	Noms de la tâche	Camille	Erwan	Victor	Mr DA SILVA
1	Organisation	A R	A	A	I
2	Environnement de travail	A	A R	A	C
3	Stats données	A	A	A R	
4	Changement d'échelle	A R	A	A	
5	Classement par densité	A	A	A R	
6	Réaliser la carte	A R	A	A	
7	Projection de la carte	A	A R	A	
8	Centrage avec projection	A	A	A R	
9	Post Mortem et livrables	A	A R	A	

Figure 3: Matrice RACI

Enfin, il ne restait plus qu'à prévoir les risques possibles dans le déroulement du projet, c'est pourquoi nous avons utilisé une matrice SWOT pour rendre compte des différents facteurs pouvant nous faire gagner du temps, ou nous en faire perdre.

	Positif (pour atteindre l'objectif)	Négatif (pour atteindre l'objectif)
Origine Interne	⦿ Maitrise partielle de Scala, Initiation aux outils de GDP	⦿ Premier Projet Industriel, mauvaise communication, vacances
Origine Externe	⦿ Professeurs et matériels à disposition, documentation en ligne	⦿ Mauvaise gestion temps, perte de données, hors-sujet

Figure 4: Matrice SWOT

## 3 Gestions des étapes

### 3.1 Étape 1

#### Problème rencontré :

Au premier abord, nous voulions lire le fichier ligne par ligne, et séparer les éléments de ces lignes par des virgules, cependant, il existait des chaînes de caractères possédant des virgules, empêchant la bonne séparation des éléments.

#### Solution :

Suite au problème de séparation des éléments ligne par ligne, il a fallu trouver une expression régulière permettant d'écartier ce problème de virgules en donnant une séparation avec ";" ou "\ ou encore ",[0-9]" qui permettrait de gérer les conflits rencontrés avec les chaînes de caractères possédant des virgules.

### 3.2 Étape 2

#### Problème rencontré :

Plusieurs calculs permettant de mesurer une distance existent, l'objectif était de sélectionner la méthode donnant la distance la plus précise possible mais aussi la moins coûteuse en calcul.

#### Solution :

Nous avons alors implémenter plusieurs des méthodes que nous avions trouvées en nous documentant. En testant sur nos données, nous avons remarqué que la méthode d'Harversine était la plus précise. Nous avons donc choisi de l'utiliser dans la suite de notre travail.

### 3.3 Étape 3

#### Problème rencontré :

Dans les statistiques, plusieurs méthodes permettaient de calculer la médiane grâce à notre structure de données. Il fallait donc, comme dans l'étape précédente, choisir la "bonne" méthode.

#### Solution :

Ici aussi nous avons selectionné plusieurs méthodes et ensuite testé sur nos données. Nous avons choisi dans un premier temps d'utiliser la fonction .sorted pour trier le tableau, pour ensuite prendre l'élément au milieu (la médiane). Cependant, nous avons

cherché pour voir s'il y avait une meilleure méthode que le .sorted, et Erwan a suggéré d'utiliser la méthode du quick select pour trier les données. Il est avéré qu'elle était plus rapide que .sorted.

### 3.4 Étape 4

Pas de problème particulier sur cette étape.

### 3.5 Étape 5

#### Problème rencontré :

Nous avons convenu au départ de compter le nombre d'aéroports dans chaque pays en parcourant notre tableau et en mettant ce résultat dans un autre tableau (avec pour chaque case du tableau un pays lui étant associé). Cependant, il était compliqué d'attribuer un nombre pour chaque pays : la liste des identifiants de "airports.dat" n'étant pas des nombres consécutifs à chaque fois...

#### Solution :

Nous avons décidé d'utiliser les HashMap de la bibliothèque scala.collection.mutable car les HashMap permettent de créer un moyen plus pratique pour accéder aux données que l'on souhaite avoir sous la main. Il faut donc déjà parcourir une fois notre tableau de données pour créer cette HashMap, cependant on accède aux éléments avec une complexité en  $O(1)$ .

### 3.6 Étape 6

#### Problème rencontré :

L'image wrapper ne fonctionnait pas sur notre version de scala (2.12.7) et les versions supportées étaient "2.9.2", "2.10.6" et "2.11.7".

#### Solution :

Pour régler ce problème de version pour l'image wrapper nous avons décidé de regarder sur GitHub pour trouver une version adéquate. [2]

### 3.7 Étape 7

#### Problème rencontré :

Après s'être documenté sur les projections conformes et équivalentes [1], il fallait transformer un couple (latitude,longitude) en (x,y) avec (x,y) les coordonnées dans la projection voulue. Cependant nous avons remarqué qu'il fallait faire une transformation linéaire sur ces coordonnées pour les placer au bon endroit dans notre image bitmap. Comment trouver ces transformations linéaires?

#### Solution :

Nous avons affiché les points de latitude -90 à 90 et longitude -180 à 180 (le but étant de quadriller toute la zone) pour avoir la forme de la carte (qui se trouvait donc en haut à gauche de l'image, à l'envers) puis nous avons trouvé la bonne transformation linéaire en essayant plusieurs solutions grâce à des fonctions helpers nous permettant de voir les étendus sur latitude et longitude.

## 3.8 Complexité et Preuves

### 3.8.1 Complexité

Soit  $n$  le nombre de lignes du fichier csv qu'on utilise dans chaque étape (airports.dat , population.csv , surface.csv) , excepté dans l'étape 6 et 7 où l'on utilisera la taille de l'image ( $p$  lignes, $q$  colonnes).

**Étape 1 :** On parcourt une fois tout le tableau ligne par ligne, et à chaque ligne on fait 6 affectations (nous comptons la

complexité en affectation puisque c'est la seule opération effectuée dans loadAirports) donc la complexité est de  $6n$  donc  $O(n)$

```
val content = bufferedSource.getLines.toArray
```

Figure 5: Parcourir ligne par ligne le fichier

**Étape 2 :** Les fonctions pour calculer une distance entre 2 points : On compte les opérations suivantes (\*,+,/)

distanceHarversine : 13 opérations

distanceSphericalLawCosines : 6 opérations

distanceEquirectangularApproximation : 9 opérations

Après avoir choisi la distance d'Harversine pour travailler sur les aéroports, on utilise la fonction distancesArray pour calculer la distance entre tous les aéroports

distancesArray : On a une double boucle for, la première sur i qui parcourt de 0 à n , la deuxième sur j qui parcourt de 0 à i-1, avec 13 opérations par double boucle , donc une complexité de :

$$11 \frac{(n - 1)n}{2}$$

, soit

$$O\left(\frac{(n - 1)n}{2}\right)$$

```
for (i <- source.indices) {
    for (j <- 0 until i) { //exclusive bound, to is inclusive tho
        result(index) = distanceHaversine(source(i)._5, source(j)._5, source(i)._6, source(j)._6)
        index = index + 1
    }
}
```

Figure 6: Double boucle

**Étape 3 :** Pour les statistiques, les recherches de distance minimum, maximum et moyenne et l'écart-type se font en  $\theta(n)$  puisqu'on parcourt toutes les lignes une seule fois.

Cependant pour la distance médiane , nous avons utilisé un algorithme de tri (quickSelect) d'une complexité de :

$$O(n \log_2(n))$$

**Étape 4 :** Pour toutes les fonctions de l'étape 4 , nous utilisons une seule boucle qui parcourt toutes les lignes du fichier airports.dat, donc nous avons une complexité en  $O(n)$ .

**Étape 5 :** La fonction loadCsv parcourt une fois le fichier qu'il a en entrée, donc on a une complexité en  $O(n)$ .

Dans la fonction Densité , on appelle la fonction loadCsv , on remplit aussi nos HashMap en  $O(n)$  chacune , on définit une hashMap avec la fonction conversionPaysversAlpha3 en  $O(n)$  , puis on fait une boucle sur la HashMap des aéroports en ne faisant qu'une affectation. Finalement, on a une complexité de  $5n$  donc en  $O(n)$ .

**Étape 6 :** Les fonctions linearTransformation et transformToXY ne font que des opérations, respectivement 6 et 5 opérations. La fonction showTrace fait une double boucle avec une première boucle de 360 éléments et une deuxième de 180 éléments, donc on a une complexité en  $\theta(360 \times 180)$ .

**Étape 7 :** La complexité qui nous intéresse dans cette étape est celle de addAllAirportsToImage

Pour toutes les projections , les fonctions transformtoXY"..." ont des complexités en  $\theta(1)$  (quelques opérations. Ainsi, dans la fonction addAllAirportsToImage on parcourt toute l'image, pour placer les points un par un.

### 3.8.2 Preuves

Toutes les fonctions ont des boucles bornées donc terminent bien. La seule à prouver est le quickSelect cependant c'est une variante du quickSort donc la démonstration est évidente.

## 4 Production finale

Voici la production de l'étape 6 :

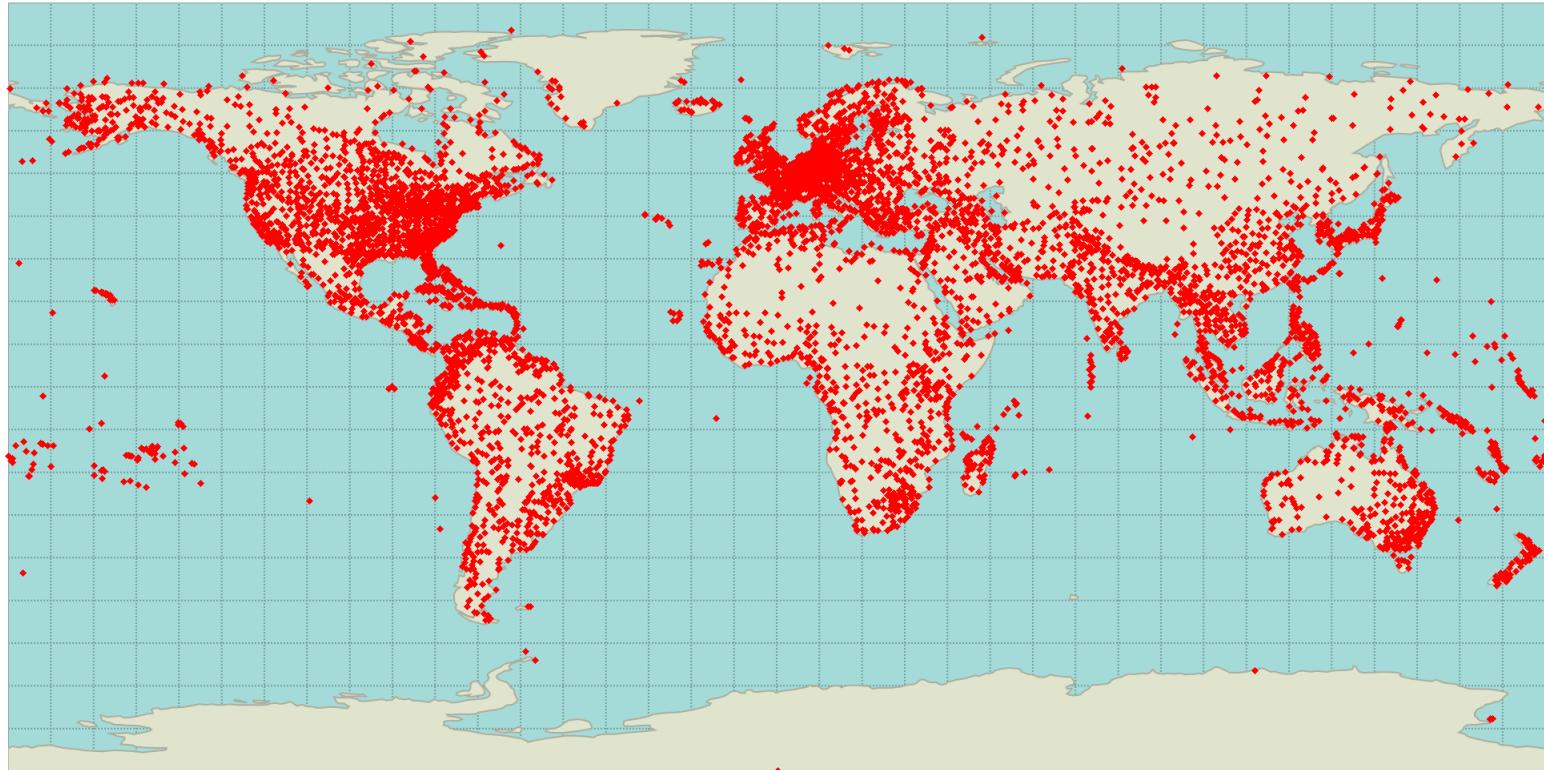


Figure 7: Étape 6

Et quelques projections :

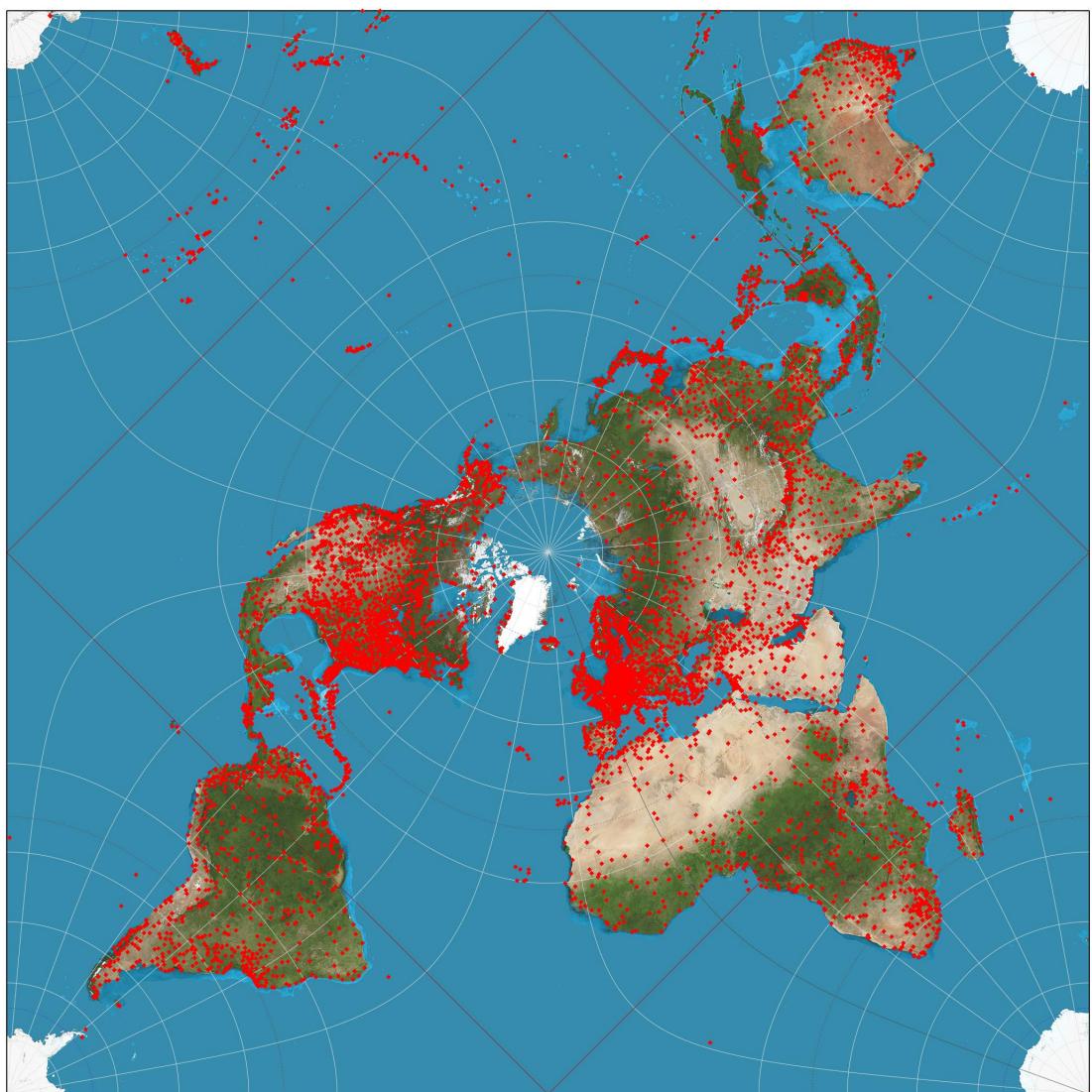


Figure 8: PierceQuincuncial Projection

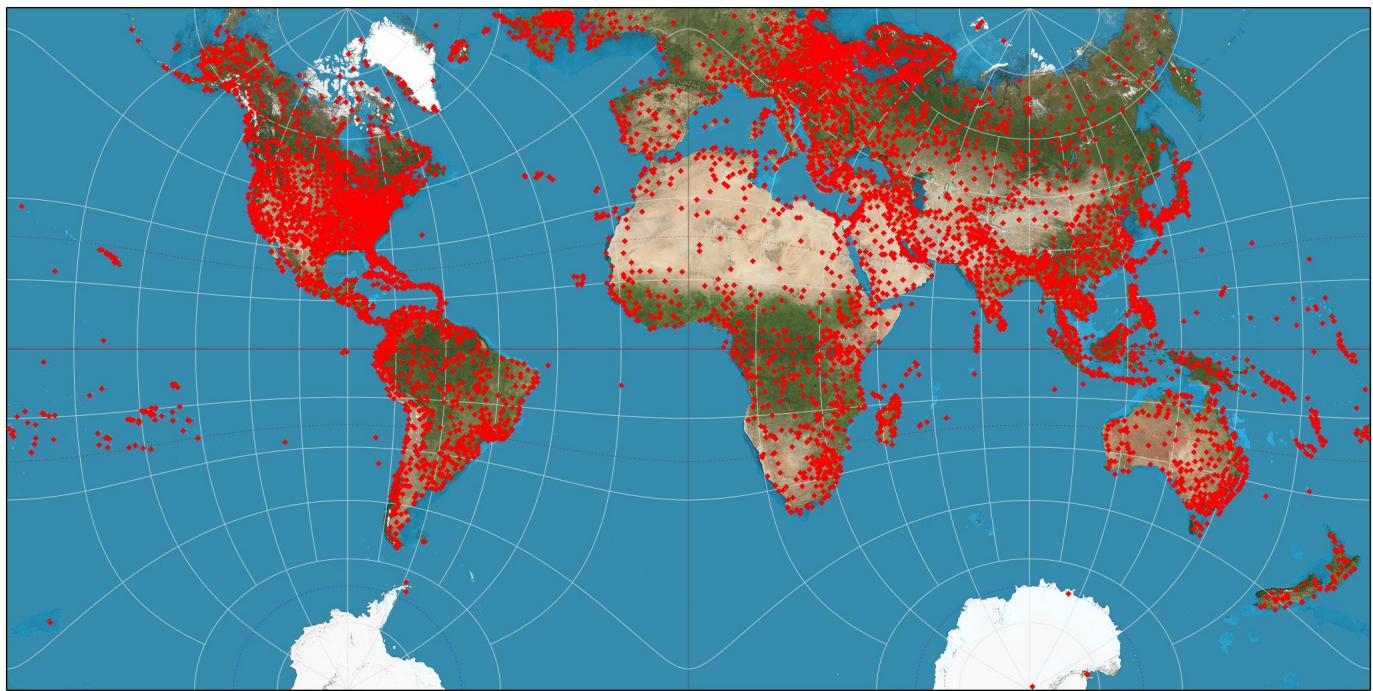


Figure 9: Guyou Projection

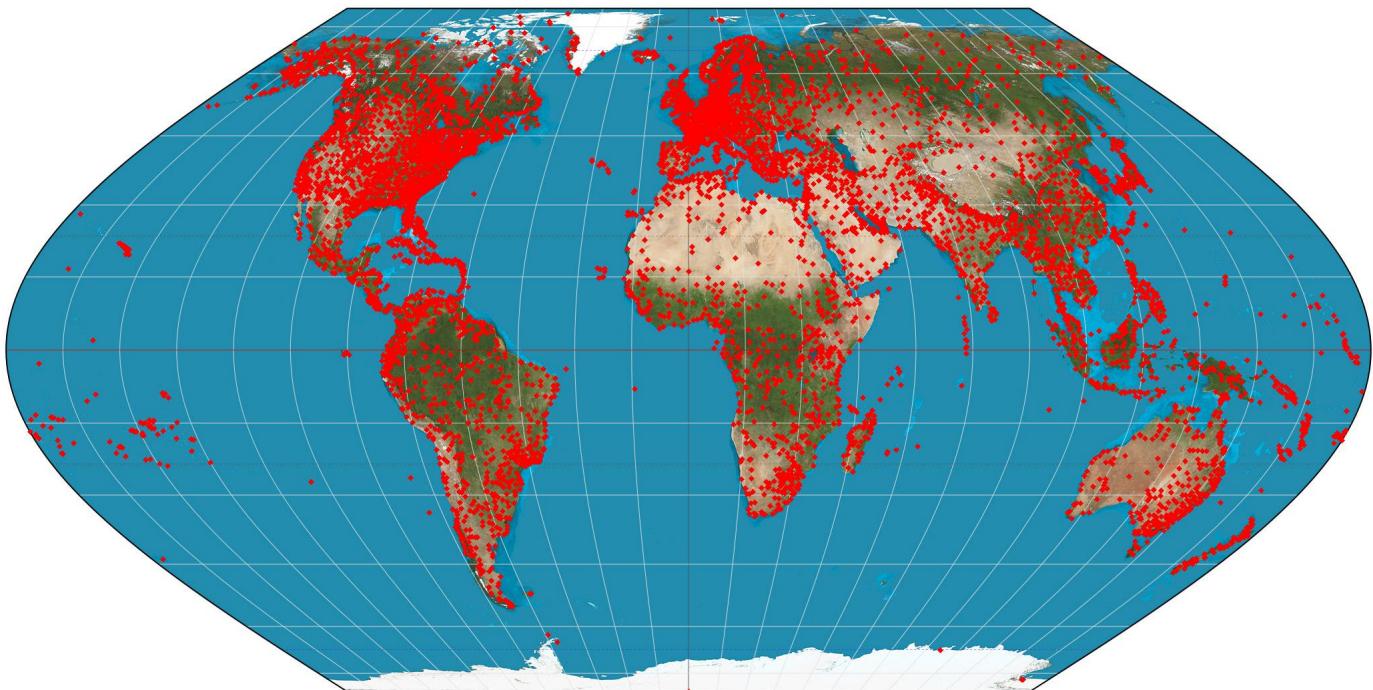


Figure 10: Eckert VI Projection

Et enfin l'interface graphique pour choisir la projection, les statistiques à afficher et le fichier csv à portée de main.

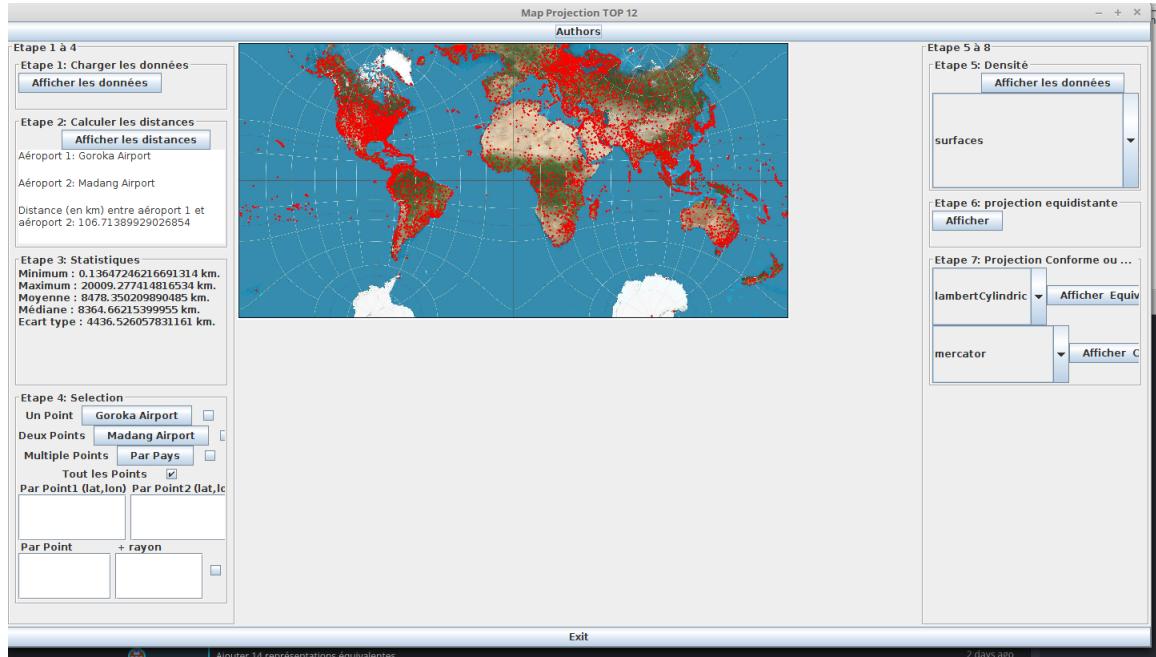


Figure 11: Interface

## 5 Ce que le projet a apporté à chacun

Tout d'abord, voici la table des heures que nous avons effectuées.

Item	Camille	Victor	Erwan
Etat de l'art	Ø	3h	10h
Organisation/GDP	5h	1h	1h
Espace de travail	Ø	Ø	3h
Documentation	7h	7h	5h
Etape 1	Ø	Ø	2h30min
Etape 2	2h	Ø	30min
Etape 3	Ø	2h	1h
Etape 4	2h	Ø	30min
Etape 5	Ø	4h	1h
Etape 6	3h	Ø	1h
Etape 7	8h	6h	14h
Etape 8	Ø	2h	40h
Test/Correction	4h	4h	6h
Complexité	5h	5h	Ø
Compte-rendus	20h	5h	Ø
Rapport	1h	20h	1h
<b>TOTAL</b>	<b>48h</b>	<b>58h</b>	<b>86h30min</b>

Figure 12: Table des heures de travail

Nous avons ensuite fait une réunion post mortem pour évaluer en 2 points ce que le projet nous a apporté : les compétences que nous avons acquises grâce à ce travail, puis les points que nous pensions devoir améliorer.

Camille :

<b>Compétences acquises</b>	<ul style="list-style-type: none"> <li>- Utilisation du HashMap (structure Hash Table scala et fonctionnement)</li> <li>- Utilisation du quickselect dans l'étape 3 pour le calcul de la médiane</li> <li>- Rédaction de compte-rendus sur latex</li> <li>- Découverte de l'outil Trello</li> <li>- Savoir utiliser git</li> </ul>
<b>Points à améliorer</b>	<ul style="list-style-type: none"> <li>- Avoir le réflexe de se documenter pour chaque interrogation</li> <li>- Utiliser des exemples pour être sûr de comprendre avant de coder</li> <li>- Se renseigner sur les outils mathématiques nécessaires avant de coder</li> </ul>

Figure 13: CamillePM

Victor :

<b>Compétences acquises</b>	<p>(Technique)</p> <ul style="list-style-type: none"> <li># Savoir utiliser les différentes librairies de scala :</li> <li>-scala.io.Source pour lire un fichier csv ligne par ligne, extraire des éléments de ces lignes.</li> <li>-scala.collection.mutable pour utiliser les HashMap qui permettent d'attribuer à des types de pointeur que l'on veut des types d'information que l'on souhaite en O(1).</li> <li># Essayer de découper le travail d'un algorithme en plusieurs petits problèmes plus facile à résoudre.</li> <li># Faire un rapport de projet et synthétiser les informations données par chacun des membres de l'équipe.</li> <li># Utiliser LateX pour faire ce rapport</li> </ul>
<b>Points à améliorer</b>	<ul style="list-style-type: none"> <li># Perdre moins de temps dans les recherches de documentation sur internet en mettant les bons mots clés.</li> <li># Se mettre plus à l'avance pour apprendre les bases d'un langage que je ne maîtrise pas.</li> </ul>

Figure 14: VictorPM

Erwan :

<b>Compétences acquises</b>	<ul style="list-style-type: none"><li>• Apprentissage du langage scala en temps record (2 semaines: JavaScript battu!)</li><li>• Apprentissage du travail en équipe et de la gestion partagé d'un git avec potentiellement plus de risque.</li><li>• Apprentissage des différentes types de projections, révision sur les fonctions elliptiques plus introduction au monde merveilleux de la géodésie</li><li>• Apprentissage de SBT (simple build tool) qui est bien plus facile que gradle même si moins documenté.</li></ul>
<b>Points à améliorer</b>	<ul style="list-style-type: none"><li>• Besoin de prendre plus de recul: par exemple sur Peirce Quincunial j'ai passé trop de temps sur les fonctions elliptiques mais cela a permis de simplifier la suite</li><li>• Apprendre à moins guider les personnes et à favoriser la créativité ainsi que le travail, certes j'ai donné un maximum d'explications mais sur un plus grand projet je devrais déléguer bien plus</li><li>• Ne pas procrastiner ! Même si cela permet de publier plus rapidement par la suite.</li></ul>

Figure 15: ErwanPM

## 6 Remerciements

- Nous souhaitons remercier l'ensemble de l'équipe pédagogique de Telecom Nancy pour nous avoir enseigné les méthodes et outils indispensable à la réalisation de notre projet.
- Plus particulièrement Mr DA SILVA (responsable du projet) et Mme HEURTEL ainsi que Rémi BACHELET pour la gestion de projet.
- Nous remercions également Telecom Nancy, pour avoir mis à notre disposition les infrastructures et le matériel informatique nécessaires au projet.

## 7 Sources

- **Sources des projections [1]** : ([https://fr.wikipedia.org/wiki/Projection\\_cartographique](https://fr.wikipedia.org/wiki/Projection_cartographique))
  - Projections conformes :
    - \* [https://en.wikipedia.org/wiki/Mercator\\_projection](https://en.wikipedia.org/wiki/Mercator_projection)
    - \* [https://en.wikipedia.org/wiki/Lambert\\_conformal\\_conic\\_projection](https://en.wikipedia.org/wiki/Lambert_conformal_conic_projection)
    - \* [https://en.wikipedia.org/wiki/Transverse\\_Mercator\\_projection](https://en.wikipedia.org/wiki/Transverse_Mercator_projection)
    - \* [https://en.wikipedia.org/wiki/Stereographic\\_projection](https://en.wikipedia.org/wiki/Stereographic_projection)
    - \* [https://en.wikipedia.org/wiki/Peirce\\_quincuncial\\_projection](https://en.wikipedia.org/wiki/Peirce_quincuncial_projection)
    - \* [https://en.wikipedia.org/wiki/Lee\\_Conformal\\_Projection](https://en.wikipedia.org/wiki/Lee_Conformal_Projection)
    - \* [https://en.wikipedia.org/wiki/Guyou\\_hemisphere-in-a-square\\_projection](https://en.wikipedia.org/wiki/Guyou_hemisphere-in-a-square_projection)
    - \* [https://en.wikipedia.org/wiki/Adams\\_hemisphere-in-a-square\\_projection](https://en.wikipedia.org/wiki/Adams_hemisphere-in-a-square_projection)

- Projections équivalentes :
  - \* [https://en.wikipedia.org/wiki/Lambert\\_cylindrical\\_equal-area\\_projection](https://en.wikipedia.org/wiki/Lambert_cylindrical_equal-area_projection)
  - \* [https://en.wikipedia.org/wiki/Behrmann\\_projection](https://en.wikipedia.org/wiki/Behrmann_projection)
  - \* [https://en.wikipedia.org/wiki/Eckert\\_projection](https://en.wikipedia.org/wiki/Eckert_projection)
  - \* [https://en.wikipedia.org/wiki/Gall-Peters\\_projection](https://en.wikipedia.org/wiki/Gall-Peters_projection)
  - \* [https://en.wikipedia.org/wiki/Hobo-Dyer\\_projection](https://en.wikipedia.org/wiki/Hobo-Dyer_projection)
  - \* [https://en.wikipedia.org/wiki/Mollweide\\_projection](https://en.wikipedia.org/wiki/Mollweide_projection)
  - \* [https://en.wikipedia.org/wiki/Sinusoidal\\_projection](https://en.wikipedia.org/wiki/Sinusoidal_projection)
  - \* [https://en.wikipedia.org/wiki/Goode\\_homolosine\\_projection](https://en.wikipedia.org/wiki/Goode_homolosine_projection)
  - \* [https://en.wikipedia.org/wiki/Tobler\\_hyperelliptical\\_projection](https://en.wikipedia.org/wiki/Tobler_hyperelliptical_projection)
  - \* [https://en.wikipedia.org/wiki/Equal\\_Earth\\_projection](https://en.wikipedia.org/wiki/Equal_Earth_projection)
- **Image Wrapper** [2] : [https://github.com/tncyttop/top-roaddetection?fbclid=IwAR36FJKJONnHDibsnBvIsHB53R1sOlaTBIIAKgKKlp\\_CkiqD8](https://github.com/tncyttop/top-roaddetection?fbclid=IwAR36FJKJONnHDibsnBvIsHB53R1sOlaTBIIAKgKKlp_CkiqD8)

## 8 État de l'art

### Définition :

Projection cartographique : Représentation d'une surface modèle (sphère ou ellipsoïde) sur un plan.  
Il en existe plusieurs types.

#### Type de projections :

- Cylindrique :

On projette l'ellipsoïde sur un cylindre qui l'englobe. Celui-ci peut être tangent au grand cercle, ou sécant en deux cercles. Puis on déroule le cylindre pour obtenir la carte.

- Pseudo-cylindrique :

In standard presentation, these map the central meridian and parallels as straight lines. Other meridians are curves (or possibly straight from pole to equator), regularly spaced along parallels.

- Conique :

On projette l'ellipsoïde sur une surface conique tangente à une ellipse ou sécant en deux ellipses. Puis on déroule le cône pour obtenir la carte.

- Pseudoconique :

In standard presentation, pseudoconical projections represent the central meridian as a straight line, other meridians as complex curves, and parallels as circular arcs.

- Azimutale :

On projette l'ellipsoïde sur un plan tangent en un point ou sécant en un cercle.

- Pseudo-azimutale :

In standard presentation, pseudoazimuthal projections map the equator and central meridian to perpendicular, intersecting straight lines. They map parallels to complex curves bowing away from the equator, and meridians to complex curves bowing in toward the central meridian. Listed here after pseudocylindrical as generally similar to them in shape and purpose.

- Stéréographique :

Le point de perspective est placé sur le sphéroïde ou l'ellipsoïde à l'opposé du plan de projection. Le plan de projection, qui sépare les deux hémisphères nord et sud de la sphère, est appelé plan équatorial

- Orthographique :

Le point de perspective est à une distance infinie. On perçoit un hémisphère du globe comme si on était situé dans l'espace. Les surfaces et formes sont déformées, mais les distances sont préservées sur des lignes parallèles.

D'autres projections sont calculées seulement avec des formules, et ne sont pas basées sur des projections particulières.

- Polyédrique :

Polyhedral maps can be folded up into a polyhedral approximation to the sphere, using particular projection to map each face with low distortion

**Propriétés :**

- Équivalente :

Conserve localement les surfaces. is constant in all directions from any chosen point.

- Conforme :

Conserve localement les angles, donc les formes.

- Aphyllactique :

On ne conserve plus de métrique , mais on essaie des réduire les distorsions.

- Equidistante :

Conserve les distances sur les méridiens.

- Gnomonique :

Transforme les grands cercles en lignes droites. Le point de perspective est au centre du sphéroïde. La projection gnomonique conserve les orthodromies, puisque tout arc de grand cercle est projeté en un segment.

**Liste de certaines projections :**

- Mercator (Conforme,Cylindrique) Google Maps utilise cette projection.

- Peters (Équivalente,Cylindrique)

- Robinson (Pseudo-cylindrique, aphyllactique)

- Mollweide (Pseudo-cylindrique)

- Albers (Conique)

- Projection conique conforme de Lambert (Conique,Conforme)

- Projection azimutale équivalente de Lambert (Azimutale,Équivalente)

On peut mélanger différentes projections, utiliser des propriétés mathématiques de certaines fonctions comme des sinusoïdes ou encore effectuer des découpages dans une projection afin de la rendre la plus fidèle possible.

Équivalente et conforme s'excluent mutuellement. Les métriques sont la surface, la forme, les angles , la distance, l'échelle. Toute projection doit s'appuyer sur un datum géodesique , pour cela il existe plusieurs ellipsoïdes courants :

- Clarke 1866

- Clarke 1880 anglais

- Clarke 1880 IGN

- Bessel

- Airy

- Hayford 1909

- International 1924
- WGS 66
- International 1967
- WGS 72
- IAG - GRS 80
- WGS 84
- NADS27
- NADS83
- OSGB36
- ETRS89
- ED50
- GDA94
- JGD2011
- Tokyo97
- KGD2002
- TWD67 et TWD97
- BJS54
- XAS80
- GCJ - 02
- BD - 09
- PZ - 09.11
- GTRF
- CGCS200
- Hong Kong Principal Datum
- ITRF2014

**Librairies existantes permettant d'effectuer des projections cartographiques :**

- C/C++ : <https://proj4.org/>
- Java : <https://github.com/OSUCartography/JMapProjLib> et <https://github.com/orbisgis/cts>
- JavaScript : [https://github.com/d3/d3-geo-projection/](https://github.com/d3/d3-geo-projection) et <http://proj4js.org/>
- Python : <https://github.com/jswhit/pyproj>, <https://github.com/geo-data/python-epsg> et <https://github.com/SciTools/cartopy>
- Go: <https://github.com/pebbe/go-proj-4> et <https://github.com/omniscale/go-proj>
- Rust : <https://github.com/georust/rust-proj>

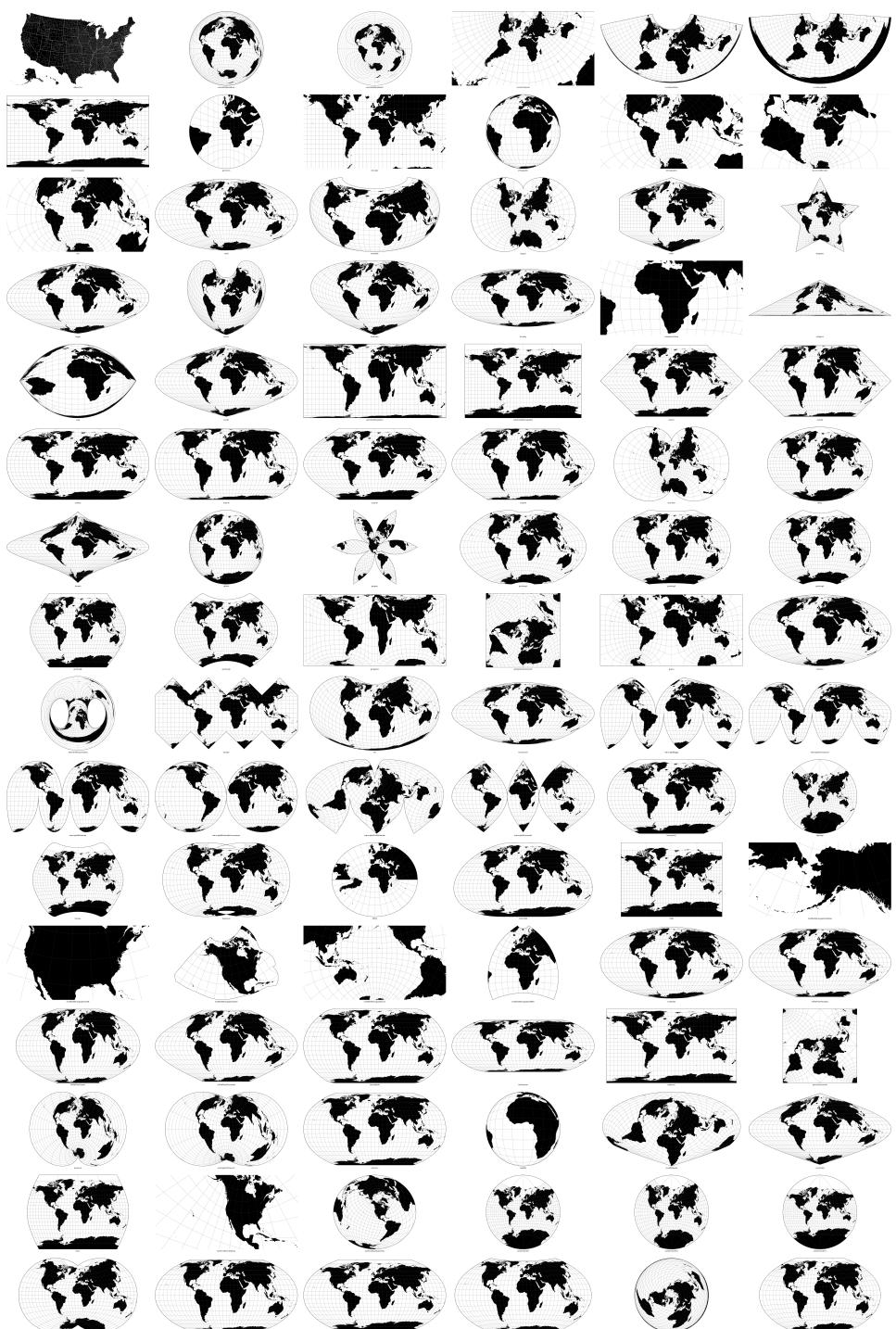


Figure 16: Images de différentes projections