OPENLAYERS 3

UNE BIBLIOTHÈQUE UNIQUE !

camptocamp

INNOVATIVE SOLUTIONS
BY OPEN SOURCE EXPERTS

Ou

« Pourquoi OL3 est fantastique ? »

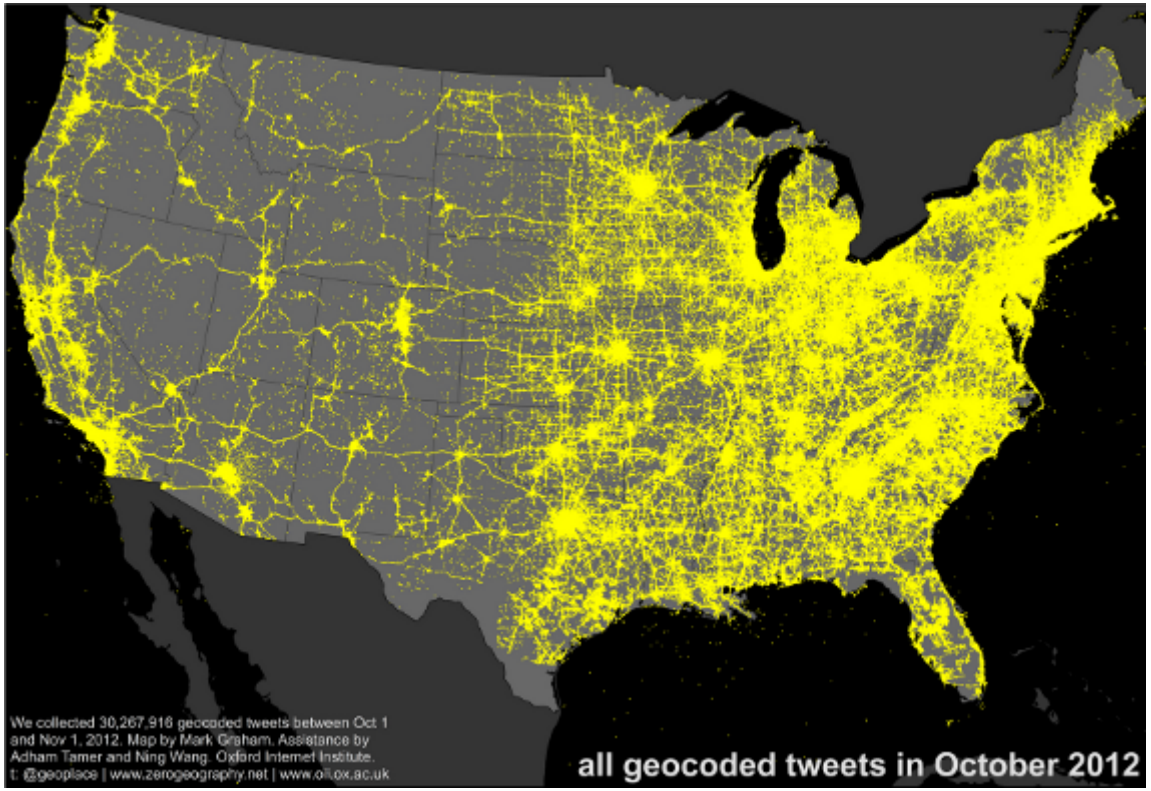
Plan

- Objectifs/Vision OL3 – rappel
- Exemples de techniques/outils utilisées dans OL3

Objectifs/Vision OL3

Convergence 2D 3D

Vecteur riche/complexe



Les cartes sont des graphiques

Résumé des objectifs

- Richesse fonctionnelle
- Gestion de données 3D
- Gestion de vecteurs complexes
- Qualité des rendus (retina)
- Performance et robustesse

« Les cartes sont des graphiques »

Techniques/outils utilisées dans OL3

Du vrai vecteur !

Les vecteurs sont dessinés **très souvent**.

- pendant les animations
- pendant les interactions (pendant le "pinch-zoom" !)

Avantages :

- qualité du rendu – pas de "blur" suite à une rotation
- les labels et icônes ne tournent pas avec la carte

La démo…

Performance

Comment dessiner souvent et obtenir de bonnes performances ?

⇨ Utilisation de techniques et algorithmes adaptés.

On pousse l'implémentation à ses limites !

Simplification des géométries

On dessine des géométries simplifiées, pour ne pas dessiner des "vertex" qui sont sur le même pixel.

- Douglas Peucker
- "Quantization" – maintien de la topologie

La simplification permet aussi un rendu de meilleure qualité aux petites échelles.

La démo…

Batching

On minimise les traitements et manipulations de données.

- calcul des styles
- simplification des géométries
- lecture des features dans le R-tree
- et toutes les implications sur le garbage collector

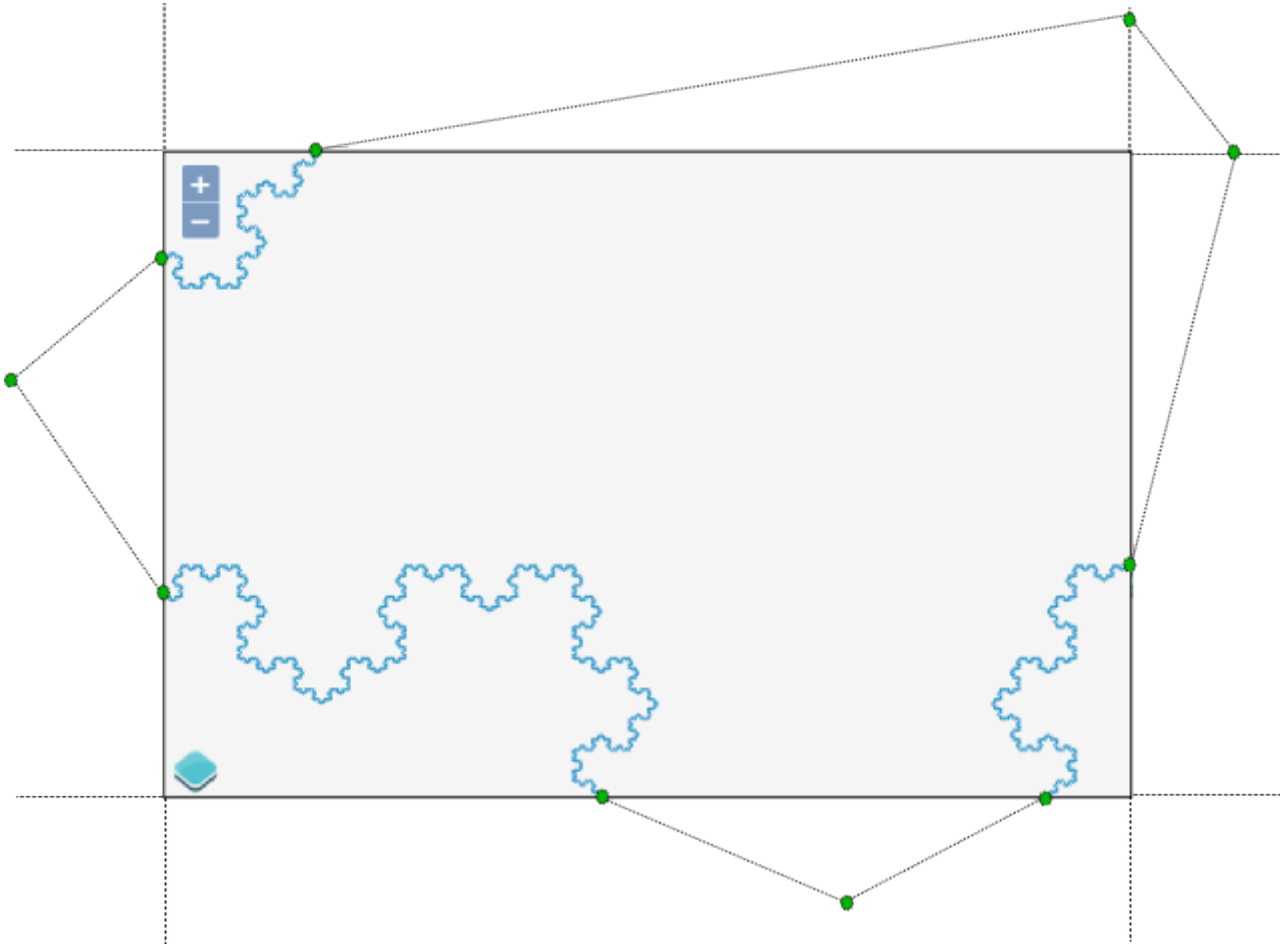
⇨ Système de "batch/replay" dans le renderer. On réutilise le batch pendant les animations et les interactions.

Important pour WebGL !

La démo...

Sur-simplification et clipping

Sur-simplification + clipping pour les parties en dehors du viewport.



La démo...

Hit Detection

Pas de "hit detection" natif avec Canvas (et WebGL).

⇨ On redessine toute la "scène" dans un canvas de 1x1 px, et on teste si on a une couleur. Réutilisation du batch.

Avantages :

- détection parfaite au pixel
- tolérance pour la détection de lignes sur device touch
- simple et efficace

La démo…

Compilateur Closure 

Outil unique dans le monde JavaScript !

- Renommage des propriétés
- Élimination du code non utilisé
- Applatissage des namespaces
- Dévirtualisation des méthodes
- "Inlining"

Exemple Compilateur Closure

```
goog.provide('ANamespace.ASubNamespace.AClass');

// une classe
ANamespace.ASubNamespace.AClass = function() {
  this.aProperty = 'prop1';
};

// une méthode
ANamespace.ASubNamespace.AClass.prototype.aMethod = function() {
  this.aProperty = 'change';
};

// une instance
var anInstance = new ANamespace.ASubNamespace.AClass();

// appel d'une méthode
anInstance.aMethod();
```

compilé en :

```
window.b=new function(){this.a="prop1"};window.b.a="change";
```


Avantages du compilateur

Gérer un gros volume de code.

- faire des petits "builds" d'OL3
- faire des "builds" combinant OL3 + application
- "type-checking"

Mais : bien utiliser le compilateur demande de l'expérience.

Conclusion

Conclusion

- Bibliothèque ambitieuse
- Techniques innovantes
- Canvas et WebGL

État actuel

- v3.0.0-beta.5
- Doc des API grandement améliorée
- Outil de build "custom" amélioré

Perspectives

- Implémentation WebGL du vecteur
- Intégration avec Cesium + 3D en général
- Outils de build en ligne
- Tutoriel : créer des appli OL3 avec Closure
- « Amélioration continue »

Merci !

As Soon As Possible!