L'air de rien
N° 10

L'aléastriel du **L**aboratoire de **R**echerche et de **D**éveloppement de l'**E**PITA[1] Numéro 10, Juin 2007

Édito

par *Alban Linard (Doctorant)*

Ce numéro suit l'actualité du G8[2], à défaut de la météo, en abordant le thème du réchauffement climatique. Il présente en effet certaines des dernières avancées de la recherche dans le domaine de la survie en milieu désertique.

Un premier article initiera le lecteur à la simulation du déplacement des Barkhanes, annonçant ainsi les futures prévisions d'itinéraires ensablés d'Oryx[3] Futé. Il est suivi d'un second article sur l'utilisation de chameaux pour sécuriser la fabrication de filets à brouillard[4] (sur le principe des toiles d'araignées).

Du sable, du vent et des dunes

par *Olivier Ricou (Enseignant-Chercheur)*

Un des sujet de l'analyse numérique consiste à simuler les écoulements. Les écoulements sont omniprésents, la météo simule les déplacements des masses d'air, l'aéronautique le profil des avions qui glisseront le mieux, d'autre regarderons l'impact d'un tsunami sur une côte et nous, nous nous concentrons sur le déplacement des dunes.

Une dune se déplace sous l'action du vent qui déplace les grains de sable, petit bond par petit bond pour donner l'impression finale du déplacement de ces montagnes de sable que sont les barkhanes, ces grandes dunes en forme de croissant qui avancent sur un sol dur, cf. sur Google Maps dans le sud Maroc autour de Laayoune[5] par exemple.

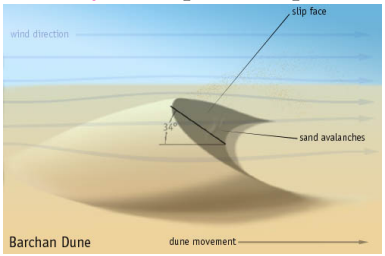

FIG. 1 – Schéma d'une Barkhane, Wikipédia 2007

Dans tous les cas, les équations de base sont les équations de Navier et Stokes :

$$\begin{cases} \rho\left(\partial_t \mathbf{u} + \mathbf{u}.\nabla\mathbf{u}\right) + \nabla p - \nabla.\mu(\nabla\mathbf{u} + \nabla\mathbf{u}^T) &= \mathbf{f} \\ \partial_t \rho + \nabla.(\rho\mathbf{u}) &= 0 \end{cases}$$

où ρ est la densité du fluide, μ sa viscosité, $\mathbf{u}$ sa vitesse, p sa pression et $\mathbf{f}$ les forces extérieures. La première équation est la conservation du mouvement, la seconde représente la variation de la masse.

Comme il n'existe pas de solution générale de ces équations, il est nécessaire de calculer une approximation sur un maillage. Pour cela on utilise différente méthodes mathématiques, comme la méthode des différences finies ou des éléments finis, toutes basées sur la discrétisation des équations sur un maillage. Cela revient à calculer les inconnues, vitesse, pression, sur une ensemble de points finis, les nœuds du maillage. Pour cela on évalue localement les équation sur chaque nœud ce qui donne à résoudre à chaque pas de temps un système matriciel de taille nombre de nœuds par nombre de nœud. Mais c'est le pire des cas et heureusement, on peut souvent simplifier comme on va voir.

Dans notre cas on a deux fluides qui se mélangent à l'interface, le vent et le sable. Il faut donc construire

[1] L'air de rien, http://publis.lrde.epita.fr/LrdeBulletin.
[2] G8, http://www.g-8.de/nn_94680/Content/EN/Artikel/__g8-summit/2007-06-07-g8-klimaschutz__en.html.
[3] Sanctuaire de l'oryx arabe, http://whc.unesco.org/fr/list/654/.
[4] Capteurs de brouillard à Chungungo, http://www.crdi.ca/un_focus/ev-26965-201-1-DO_TOPIC.html.
[5] Barkhanes, http://maps.google.fr/?ie=UTF8&om=1&z=15&ll=27.463309,-13.144541&spn=0.031491,0.044761&t=h.

un maillage suffisamment fin pour saisir les variations locales à l'origine des phénomènes de plus grand échelle, et assez grand pour avoir la vision globale. Dans le cas des dunes, le mouvement de la dune est lié à la reptation des grains de sable ce qui impose d'avoir un maillage assez fin pour saisir ce mouvement, donc la taille des mailles doit être de l'ordre de 10^{-4} mètres. En même temps il faut que le maillage englobe la dune qui fait quelques mètres de haut et une centaines de largeur, donc un domaine en 10^2. Nous sommes en 3 dimensions donc un maillage régulier aurait environ $10^{6^3} = 10^{18}$ nœuds[6]. Même si la matrice d'un tel problème est très creuse et même si on peut faire un maillage dont la densité varie, il n'est pas réaliste actuellement d'espérer résoudre directement un tel système.

Ce problème est un problème d'échelle où le trop petit et le trop grand se mélange. Dans ce cas une solution consiste à trouver des équations qui permettent de simuler correctement les phénomènes de petite échelle pour ne plus avoir à mailler trop finement. Dans notre cas, on va aller plus loin puisqu'on va ramener ce problème à une série de petits problèmes en 1 dimension.

Pour commencer on profite du fait que les barkhanes se forment dans des endroits ayant un vent unidirectionnel. Qui dit direction privilégiée, celle du vent, dit possibilité de pouvoir résoudre le problème tranche par tranche. D'un point de vue numérique cette approche est très agréable puisque permet de diminuer sérieusement la taille du problème, malheureusement physiquement cette approximation est grossière. En particulier si tous les grains de sable avançaient uniquement dans la direction du vent, il ne pourrait pas y avoir création des cornes de la barkhane. Le profil des dunes devrait être le même, comme les vagues, cf. toujours sur Google Maps[7]. Il a donc fallu trouver un compromis, une approximation numérique qui utilise ce découpage en tranches mais avec une formulation qui permette de faire passer des grains de sable d'une tranche à l'autre. Bien sûr cette formulation doit être assez légère pour que le gain du découpage en tranches ne soit pas annulé.

Une autre façon de simplifier le problème est de regarder ce qui se passe verticalement. Si on va sur le terrain, on constate que prêt du sol l'air est saturé de sable alors que plus haut, ça va mieux. On constate aussi que l'air n'est pas toujours saturé. Par exemple si on se met dans le creux de la barkhane, il n'y a pas de sable qui vole, il y en a un peu au vent de la dune et l'air en est saturé au sommet ou sous le vent des cornes. Comme le mouvement de la dune est dû au déplacement des grains de l'arrière vers l'avant, ce flux de sable semble donc un indicateur naturel. On va donc reprendre nos équations et remplacer la vitesse des fluides par le flux de sable $\mathbf{q}$:

$$\mathbf{q}(x,y) = \int_{z=h(x,y)}^{\infty} \rho(x,y,z)\,\mathbf{u}(x,y,z)\,dz$$

où h est la hauteur du sable au dessus du sol (donc zéro si on n'est pas sur la barkhane) et où la densité l'air ρ varie en fonction de la quantité de sable transportée.

Ainsi on peut intégrer verticalement l'équation de variation de la masse, ce qui donne :

$$\begin{aligned} \int_0^{\infty} \partial_t \rho + \nabla.(\rho\,\mathbf{u})\,dz &= 0 \\ \partial_t \int_0^{\infty} \rho\,dz + \int_{h(x,y)}^{\infty} \partial_x \rho u_x + \partial_y \rho u_y\,dz + [\rho\,u_z]_0^{\infty} &= 0 \\ \partial_t\, h + \nabla_{2D}.\mathbf{q} &= 0 \end{aligned}$$

sachant que la vitesse verticale, u_z, est nulle au sol et à l'infini et que la quantité de sable entre le sol et le ciel est, à très peu près, la hauteur de la dune.

On a ainsi transformé notre problème 3D initial où les inconnues étaient la vitesse du fluide et sa pression en un problème mieux adapté à notre problème 2D où les inconnues sont la hauteur de la dune et le flux de sable transporté par le vent. De plus, en utilisant un terme qui indique quelle quantité de sable passe d'une tranche à une autre, et donc en faisant varier le flux, on peut ramener le problème en N problèmes 1D dont la première équation est

$$\partial_t\, h + \partial_x q = 0$$

Il faut maintenant une seconde équation pour résoudre notre problème à deux inconnues ce qu'on fera au prochain numéro. En attendant vous pouvez déjà aller voir une simulation qui montre la création d'une barkhane à partir d'un tas de sable[8]. Vous noterez les petites barkhanes qui se créent lorsque des paquets de sable se détachent. Vous noterez aussi que les petites barkhane avance bien plus vite que les grandes ce qui explique les nombreuses collisions qu'on peut observer sur les photos aériennes.

[6] Un tel maillage ne pourrait pas tenir sur l'ensemble des disques dur de l'EPITA.

[7] Vagues de dunes, http://maps.google.fr/?ie=UTF8&om=1&z=11&ll=27.811142,11.635895&spn=0.462177,0.845261&t=h.

[8] Simulation de création de barkhane, http://www.lrde.epita.fr/~ricou/creationBarkhane.avi.

Framework Web en Caml

par *Quentin Hocquet (Étudiant CSI 2008)*

Le développement Web étant toujours en plein essor, de nombreux framework sont utilisés : PHP, J2EE, Ruby on Rails, frameworks en Python, ... Leur très grande majorité utilise des langages de scripts typés dynamiquement, et le plus souvent très faiblement. Il est communément admis qu'il est acceptable de sacrifier le typage et la sécurité qu'il apporte pour gagner en simplicité et en rapidité de développement.

Pourtant, il est possible de créer des systèmes très fortement typés et sécurisés, et néanmoins complets et simples.

Donnez moi de la syntaxe concrète

Par opposition à la syntaxe abstraite, qui fait abstraction de la mise en page pour ne conserver que la sémantique (comme dans un AST[9]), la syntaxe concrète représente le code tel qu'un programmeur l'écrit.

Dans le cadre de la transformation de programme, l'utilisation de syntaxe concrète consiste à embarquer un langage dans un autre de façon transparente. Deux langages peuvent cohabiter dans un même fichier, avec leur syntaxe d'origine.

Grâce à camlp4[10], il est possible d'embarquer du HTML, du SQL, du CSS et du JavaScript dans notre code Caml.

Simplicité et validité

La syntaxe concrète permet d'avoir un style clair et concis, qui n'a rien a envier à des langages de script tels que PHP.

```
let index args =
  <<
    <html>
    <head><title>Hello world!</title></head>
    <body><p>C'est fou !</p></body>
    </html>
  >>

let _ =
  MlWeb.run index
```

Si l'on a conservé toute la simplicité et la lisibilité d'un langage de script, on a également gagné énormément en validité et en robustesse. En effet, nous avons la certitude *à la compilation* que le HTML généré sera valide. Les langages de script basés sur la concaténation de chaînes ne fournissent aucune garantie à ce niveau.

Ici, les inclusions de HTML ou de chaînes de caractères sont également typées. L'exemple suivant affiche les arguments de la page sous forme de liste :

```
let convert (name, value) =
  << <li>L'argument $name vaut $value.</li> >>

let index args =
  let list =
    map convert args
  in
    << <ul> ~list </ul> >>
```

Sécurité

La sécurité est capitale sur Internet, mais pourtant les outils utilisés sont remplis de pièges subtils. Les langages basés sur la concaténation de chaînes sont extrêmement vulnérables aux attaques par injection, du fait qu'il sont extrêmement peu typés. Un utilisateur malveillant peut, par exemple, faire une injection JavaScript dans un code HTML, et rediriger les visiteurs vers sa propre page. Il faut, dans ces langages, penser à vérifier toutes les données dynamiques pour prévenir ces injections, ce qui est une source considérable de bugs et de failles.

L'utilisation de syntaxe concrète fortement typée rend tout simplement *impossible* toute forme d'injection. Considérons le code suivant, qui transforme une donnée entrée par l'utilisateur en HTML a afficher :

```
let show user_string =
  << <p>Vous avez entré : $user_string.</p> >>
```

Dans un langage de script comme PHP, il s'agirait d'une fonction critique. Il faudrait penser a échapper les caractères spéciaux dans la chaîne récupérée, pour l'empêcher d'injecter du JavaScript par exemple. En Caml, grâce au typage fort, le framework sait qu'il s'agit d'une chaîne, et tous les caractères spéciaux, comme les fameux chevrons, seront échappés. Ces outils évitent ainsi automatiquement une des failles les plus courantes sur le Net.

Accès typé statiquement aux bases SQL

Grâce à la syntaxe concrète et a un interpréteur SQL, il devient possible d'interroger une base SQL en ayant la garantie *à la compilation* que les requêtes sont valides. À l'aide d'un fichier de méta-données, l'existence des tables, des colonnes, les types, et bien d'autres choses peuvent être vérifiées. Le retour des requêtes est également automagiquement typé.

[9] Abstract Syntax Tree
[10] Caml Preprocessor and Pretty Printer

```
let who = "Knuth"
let foo = Sql.query
  << SELECT id FROM users WHERE name = $who>>

let bar = Sql.query
  << SELECT age FROM users WHERE name = 51>>
(* characters 59-64:
this expression has type int
but is here used with type string *)
```

Cette approche surpasse à nouveau largement celle de la construction des requêtes sous forme de chaînes. Tout d'abord, il est garanti que la requête sera syntaxiquement correcte, alors qu'une erreur triviale est vite arrivée en manipulant des chaînes. D'autre part, tout comme pour le HTML, il est ici *impossible* d'être la cible d'une injection SQL, qui est probablement la faille la plus courante et la plus grave sur le Web.

D'autres framework (Java, RoR[11]) créent un modèle objet pour accéder aux bases de données. Cette approche évite également les injections, mais reste inférieurs à l'utilisation de la syntaxe concrète, car elle prive l'utilisateur de nombreuses construction complexes et puissantes en SQL, telles que `GROUP BY`, `EXISTS`, les jointures multiples, etc.

```
if Sql.query << SELECT EXISTS ( ... ) >> then
  Sql.query << SELECT birthday FROM users
    LEFT OUTER JOIN profiles ON ... >>
```

Pour conclure

Voici un exemple des bénéfices d'un typage fort et de la syntaxe concrète pour le développement Web. Dans ce véritable melting-pot des langages, cette approche offre également d'autres perspectives intéressantes : permettre d'effectuer automatiquement des transformations sur des documents HTML, ou encore embarquer CSS et JavaScript, pour permettre de créer des interfaces hautement modulaires et dynamique, grâce notamment à Ajax[12].

D'autre part, le framework possède déjà bien d'autres fonctionnalités, et des bibliothèques permettant de gérer de nombreux aspect récurrents dans la création de sites Web.

En bref

Les collaborations

Réda Dehak est accueilli actuellement, pour la durée d'un mois, au Centre de Recherche Informatique de Montréal[13] dans le cadre d'une collaboration sur le développement des systèmes de vérification du locuteur basés sur les méthodes à noyau (SVM).

Les nouvelles publications (disponibles sur publis.lrde.epita.fr)

- **Ricou, O.**. 10 years of confrontation between french internet users and their successive governments. In *Proceedings of the 7th European Conference on e-Government (ECEG)*
- **Dehak, R., Dehak, N., Kenny, P., and Dumouchel, P.**. Linear and non linear kernel GMM supervector machines for speaker verification. In *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, Antwerp, Belgium
- **Baillard, A., Berger, Ch., Bertin, E., Géraud, Th., Levillain, R., and Widynski, N.**. Algorithme de calcul de l'arbre des composantes avec applications à la reconnaissance des formes en imagerie satellitaire. In *Proceedings of the 21st Symposium on Signal and Image Processing (GRETSI)*, Troyes, France

Republished **Verna, D.**. How to make Lisp go faster than C. *IAENG International Journal of Computer Science*, 32(4)

Les logiciels

Vaucanson 1.0a [14] L'équipe Vaucanson est fière d'annoncer la sortie de Vaucanson 1.0a, une version qui, principalement, corrige des bugs et ajoute diverses optimisations.

FiNK 2.0 [15] Introduction de nouvelles macros découpant les composantes du nom de fichier courant.

[11] Ruby on Rails
[12] Asynchronous JavaScript and XML
[13] CRIM, http://www.crim.ca/fr/index.html.
[14] Vaucanson 1.0a, http://vaucanson.lrde.epita.fr/Vaucanson100a.
[15] FiNK 2.0, http://www.lrde.epita.fr/~didier/comp/development/software.php#fink.