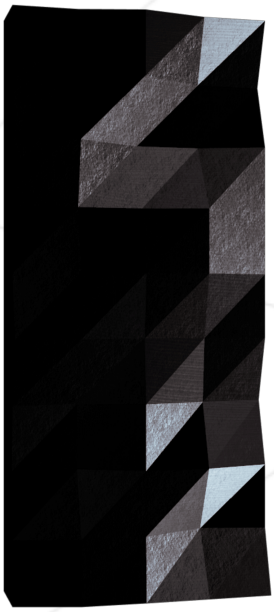


H42N42

English version coming soon !

Résumé: *Ce projet a pour but de faire découvrir les rudiments de la programmation Web côté client en* `OCaml` *grâce à Ocsigen.*

Table des matières

Chapitre I

Préambule

Depuis sa création le Web a beaucoup évolué. Au tout début, il était composé uniquement de pages statiques (documents html liés entre eux). Rapidement ces pages ont été complétées par des pages générées côté serveur à partir d'une base de données (Wikipedia, vente en ligne, journaux en lignes, forums...). Plus récemment, le Web a subi une transformation radicale en devenant une plateforme capable de faire tourner de véritables applications (traitements de texte, jeux,...), dont une partie du calcul se déroule sur le serveur et une autre dans le navigateur.

Les langages Web des années 90 (`PHP`, `Javascript`...) n'avaient pas été conçus pour ce genre d'applications. La plupart d'entre eux sont des langages de scripts n'offrant aucune garantie statique de bon fonctionnement (pas de typage statique), ce qui rend le débogage fastidieux, la maintenance du code difficile, et qui laisse passer beaucoup de failles de sécurité.

Par ailleurs, les développeurs Web sont la plupart du temps obligés de jongler entre plusieurs langages : `Javascript` côté client, `PHP`, `Ruby` ou `Python` par exemple côté serveur, `SQL` pour la base de données, etc. Cela implique de convertir les données entre ces langages et de réécrire certaines fonctions plusieurs fois si l'on veut pouvoir les exécuter des deux côtés, ce qui introduit de nombreuses sources d'erreurs pouvant introduire d'autres problèmes de sécurité.

Le projet de recherche `Ocsigen` s'est donné comme objectif d'inventer une nouvelle façon de programmer des applications Web modernes, en mettant l'accent sur l'*expressivité* et la *fiabilité*. Le projet a abouti à la création d'un framework complet distribué comme logiciel libre contenant un compilateur, un serveur Web et de nombreuses bibliothèques. `Ocsigen` s'appuie sur les nombreux concepts de haut niveau présents dans le langage `OCaml` (variants polymorphes, clôtures, types algébriques généralisés, etc.) pour écrire des applications complexes en peu de lignes de code. Et surtout, `Ocsigen` tire partie du puissant système de types d'`OCaml` pour vérifier de nombreuses propriétés du programme au moment de la compilation. Il va même jusqu'à vérifier au moment de la compilation que vos programmes ne peuvent pas générer des pages qui ne respectent pas les recommandations du `W3C`, ou dont les formulaires ne correspondent pas aux services vers lesquels ils pointent par exemple.

Une des caractéristiques principales d'`Ocsigen` est de permettre de programmer la totalité de l'application (serveur et client) en `OCaml`, dans un seul et même code. Au moment de la compilation, les portions devant être exécutées dans le navigateur sont extraites et traduites en `JavaScript` très efficace par le compilateur `Ocsigen Js_of_ocaml`.

`Ocsigen` est développé par des laboratoires de recherche du `CNRS`, de l'université Paris Diderot (P7), de l'`Inria` (Institut National de Recherche en Informatique et Automatique) et par l'entreprise BeSport, qui développe un réseau social sur le thème du sport.

`Ocsigen` est déjà utilisé par des entreprises pretigieuses dans le monde, notamment par l'université de New York (CSGB Genomics Core), par des startups, des projets open source ou des laboratoires de recherche. `Facebook` par exemple, utilise le compilateur `Js_of_ocaml` pour faire exécuter son code `OCaml` dans des navigateurs.

`Ocsigen` s'inscrit dans une tendance générale visant à améliorer les techniques de programmation Web par des vérifications statiques. Certains de projets tentent par exemple de rajouter du typage statique (basique) sur des langages existants, comme `Microsoft TypedScript` et `Facebook Flow` pour `JavaScript`, `Facebook Hack` pour `PHP` ou `Facebook Infer`, pour `C`, `C++` et `Java`. D'autres frameworks récents font de plus en plus souvent le choix de langages modernes fortement typés, comme `OCaml`, `Scala` (utilisé chez `Twitter`) ou `Haskell`.

ocsigen

be sport

université PARIS DIDEROT PARIS 7

Inria INVENTEURS DU MONDE NUMÉRIQUE

cnrs CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Chapitre II

Introduction

Depuis des millénaires, les Bestioles vivent dans une contrée paisible bordée par une rivière. Hélas, depuis quelques temps, la rivière est polluée par le méchant virus mortel et très contagieux H42N42. Les Bestioles ne peuvent pas s'en sortir sans votre aide ! Votre but est de les aider à rester éloignées de la rivière et d'emmener les malades à l'hôpital où elles pourront être soignées pour éviter de contaminer les autres.

Dans ce projet, vous écrirez un simulateur interactif du monde des Bestioles avec une interface Web côté client écrite en `OCaml`.

Chapitre III

Objectifs

`Ocsigen` permet de gérer tous les aspects de la programmation d'applications Web, et notamment de créer des sites et des applications client-serveur. Pour commencer doucement, ce premier projet de notre série a pour but de vous faire découvrir uniquement la programmation côté client avec la bibliothèque Eliom et le compilateur Js_of_ocaml, et notamment vous apprendre :

1. à faire tourner un programme `OCaml` dans un navigateur grâce au compilateur `Ocsigen Js_of_ocaml` ;
2. à faire valider statiquement les pages `HTML` que vous générez, pour interdire à votre programme de créer des pages invalides ;
3. à intéragir avec le DOM du navigateur et à appeler des méthodes `Javascript` à l'aide d'un programme `OCaml` ;
4. à programmer des handlers pour des événements de la souris ;
5. à programmer avec des threads coopératifs en style monadique avec `Ocsigen Lwt`.

Chapitre IV

Consignes générales

Le projet est à réaliser entièrement en `OCaml` dans sa dernière version disponible, avec le compilateur `Js_of_ocaml` et la bibliothèque `Eliom` du projet `Ocsigen`, eux aussi dans leurs dernières versions disponibles.

Vous pourrez trouver la documentation d'Ocsigen sur le site du projet. Prenez le temps de lire les tutoriels avant de commencer.

- Chaque Bestiole doit être commandée par un thread `Lwt`.
- Elle sera affichée sous la forme d'un élément du DOM (un `<div>` ou `<img>` par exemple).
- Les événements de souris doivent être programmés à l'aide du module `Lwt_js_event` de `Js_of_ocaml`. Vous trouverez des exemples d'utilisation de ce module dans les tutoriels sur le site d'`Ocsigen`.

Conseils :

- Ayez toujours sous la main les outils de débogage du navigateur ;
- Inscrivez-vous aux listes de diffusions `OCaml` et `Ocsigen` pour intéragir avec la communauté ;
- Suivez le canal IRC `#ocsigen` sur `irc.freenode.net` ;
- Lisez les `tutoriels Ocsigen`.

Chapitre V

Partie obligatoire

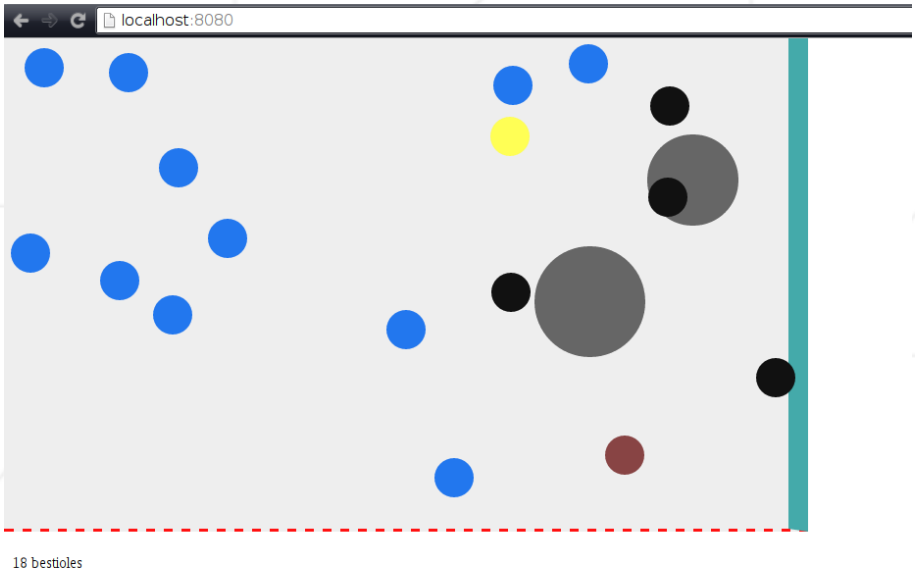
Les Bestioles doivent se déplacer de manière aléatoire sur un espace rectangulaire de la fenêtre du navigateur, bordé en haut par la rivière empoisonnée, et en bas par l'hôpital où seront soignées les malades.

Les Bestioles malchanceuses qui touchent la rivière tombent malade. Dans ce cas, elles changent de couleur, deviennent contagieuses et contaminent celles qu'elles touchent avec une certaine probabilité.

Le but du jeu est de tenir le plus longtemps possible avant que toutes les malheureuses Bestioles ne soient contaminées. Dans ce cas, le jeu est perdu et la partie s'arrête.

- Les Bestioles se reproduisent par génération spontanée tant qu'il reste au moins une Bestiole en bonne santé. La durée de leur vie est infinie, en l'absence de maladie.
- Il doit être possible de cliquer-glisser les Bestioles avec la souris pour les éloigner de la rivière ou emmener les malades à l'hôpital.
- Une Bestiole contaminée qui touche l'hôpital par hasard n'est pas soignée. Seules les Bestioles déposées sur l'hôpital par le joueur sont soignées.
- Une Bestiole en train d'être déplacée par le joueur est "invulnérable" et ne peut pas être contaminée si elle est saine.
- Une Bestiole se déplace en ligne droite mais change parfois de direction aléatoirement. Ces changement de direction inattendus ne doivent pas arriver trop souvent pour permettre au joueur de prévoir un minimum les trajectoires, mais suffisamment pour créer de la surprise.
- Une Bestiole qui touche un bord de la zone de jeu rebondit de fa con réaliste, quelque soit ce bord.
- Une Bestiole qui devient contaminée change brusquement de couleur et sa vitesse diminue de 15%.

- Une Bestiole contaminée qui touche une autre Bestiole saine a 2% de chance de la contaminer à chaque itération. Il n'y a pas de collisions (au sens "rebond"), les Bestioles passent les unes par dessus les autres.
- Une Bestiole qui devient contaminée a 10% de chances de devenir berserk ! Dans ce cas, la Bestiole a une couleur différente des simples contaminées, et son diamètre augmente doucement, jusqu'à quadrupler à la fin de sa vie, ce qui rend encore plus probables les contaminations de ses voisines...
- De la même façon, une Bestiole qui devient contaminée a 10% de chances de devenir méchante ! Dans ce cas, la Bestiole a une couleur différente des simples contaminées et des berserks, sa augmente de 15%, mais elle se met à poursuivre les autres Bestioles saines pour les contaminer volontairement ! Ouuuh la vilaine Bestiole !
- Une Bestiole contaminée ne pas pas être berserk et méchante à la fois.
- À cause de la panique, plus le temps passe, plus la vitesse de base des Bestioles doit augmenter afin de provoquer une lente mais régulière progression de la difficulté et déborder le joueur. Vous êtes libres d'utiliser la formule qui vous convient pour obtenir l'équilibrage qui vous semble le plus adapté. Le but est d'obtenir un jeu dont les premiers instants sont très faciles pour mettre le joueur en confiance mais qui dérive implacablement vers un cauchemar de clics précis et précipités.
- Le nombre de Bestioles dans la surface de jeu est laissé à votre jugement afin de vous permettre de réaliser l'équilibrage qui vous convient le mieux en fonction de la taille que vous donnerez à vos Bestioles. Évitez trop de petites Bestioles qui rendraient le jeu inintéressant.
- Il est toléré que vous adaptiez les probabilités données dans ces consignes si et seulement si ces changements servent l'équilibrage et ne dénaturent pas le jeu.
- Quand il ne reste aucune bestiole vivante, un message indique la fin du jeu.


localhost:8080
18 bestioles

Chapitre VI

Partie bonus

Lorsqu'on a investi du temps sur un projet et que le résultat est au rendez-vous, il est naturel d'avoir envie d'aller plus loin ! Bien entendu, la partie bonus n'est accessible que si et seulement si la partie obligatoire a été réalisée entièrement et parfaitement.

Voici une liste d'extensions possibles pour ce projet :

- Un bel habillage graphique pour le jeu ;
- Ajouter un moyen de contrôler les différents paramètres de la simulation à l'aide de curseurs ou de formulaires dans la page ;
- Optimiser la détection de collision entre Bestioles, en utilisant par exemple un quadtree.

Chapitre VII

Rendu et peer-évaluation

Rendez-votre travail sur votre dépot `GiT` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance.