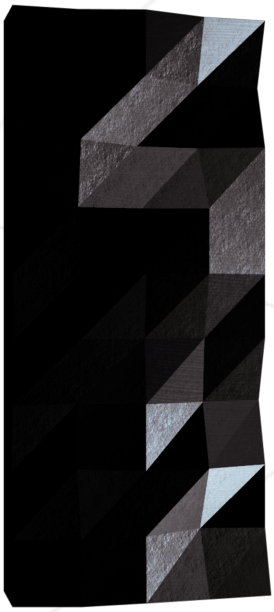


# H42N42

## Simulate the life of a population of creatures threatened by a terrifying virus

*Summary: This project aims to help you discover the basics of web programming on the client side in `OCaml`, thanks to Ocsigen.*

*Version: 5.1*

# Contents

# Chapter I

# Foreword

Since its creation, the WWW has evolved a lot. In the beginning, it was just made of static pages (html documents linked together). These pages have quickly been completed by pages that were generated on the server side from a database (Wikipedia, online store, online papers, forums...). More recently, the WWW has known a radical mutation when it became a platform that could run real applications (word processors, games...) which calculation took place in part on the server and in the browser.

The 90's web languages (`PHP`, `JavaScript`...) were not designed for this kind of application. Most of them are script-based languages that offer no proper static guarantees (no static typing), making debugging tedious, code maintenance difficult, and leaving the system open to many security breaches.

Moreover, web developers often have to handle several languages: `JavaScript` on the client side, `PHP`, `Ruby`, or `Python` – to name just a few – on the server side, `SQL` for the databases, etc. They have to convert data from one language to another, rewrite functions multiple times if they want to use them on both sides, and this, of course, creates new hazards that introduce additional security problems.

The research project `Ocsigen` aims to create a new way to program modern web applications, focusing on *expressiveness* and *reliability*. The project was a success thanks to the creation of a complete framework, distributed as open source software, containing a compiler, a web server, and many libraries. `Ocsigen` is based on several high-level concepts found in the `OCaml` language (polymorphic variants, closures, generalized algebraic types, etc.) to write complex applications with very few lines of code. More than anything, `Ocsigen` takes advantage of powerful systems like `OCaml` to check many properties of the program during compilation. It can even verify, at compile time, that your programs will not generate pages that violate `W3C` recommendations, or whose forms do not match the services they target, for instance.

Mainly `Ocsigen` allows to program the application globally (server and client) in `OCaml` in the same code. During compilation, the parts that should be executed by the browser are extracted and turned into very efficient `JavaScript` by the compiler `Ocsigen Js_of_ocaml`.

`Ocsigen` is developed by the `CNRS`, University of Paris Diderot (P7) and `Inria` (IT and Automatic Research National Institute) research labs and by the company BeSport, that develops a sports focused social network.

`Ocsigen` is already used by distinguished companies in the world, such as the University of New York (CSGB Genomics Core), by startups, open source projects or research laboratories. `Facebook`, for instance, uses the `Js_of_ocaml` compiler to execute their `OCaml` code in the browsers.

`Ocsigen` follows a general trend aimed at improving web programming techniques through static checking. For instance, some examples attempt to add basic static typing to existing languages, such as `Microsoft TypeScript` and `Facebook Flow` for `JavaScript`, `Facebook Hack` for `PHP`, or `Facebook Infer` for `C`, `C++`, and `Java`. Other recent frameworks increasingly adopt type-based languages, such as `OCaml`, `Scala` (used by `Twitter`), or `Haskell`.

ocsigen

be sport

université PARIS DIDEROT PARIS 7

Inria INVENTEURS DU MONDE NUMÉRIQUE

cnrs CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

# Chapter II

# Introduction

For millenia, Creatures have lived in a peaceful land bordered by a river. Unfortunately, the river has been polluted by H42N42, a deadly and very infectious virus for some time. The Creatures will not survive without your assistance! Your goal will be to help them stay away from the river and to bring the sick ones to the hospital where they will be healed so they don't contaminate the others.

In this project, you will write an interactive simulator of this Creature world with a Web UI written in `OCaml` on the client side.

# Chapter III

# Objectives

`Ocsigen` allows you to manage every aspect of Web app programming, and to create client-server websites and applications. For a gentle start, this project will help you discover client-side programming with the library Eliom and the compiler Js_of_ocaml, and will teach you to:

1. Run an `OCaml` program in a web browser thanks to the `Ocsigen` compiler. `Js_of_ocaml` ;
2. Statically validate the `HTML` pages that you generate to prevent your program from creating invalid pages;
3. interact with the browser's DOM and call `Javascript` methods thanks to the `OCaml` program;
4. program handlers for mouse events;
5. program cooperative threads in monadic style with `Ocsigen` `Lwt`.

# Chapter IV

# General instructions

The project will be entirely programmed with the latest version of `OCaml`, the `Js_of_ocaml` compiler and the `Eliom` library of the `Ocsigen` project. You will also need the latest version available.

You will find the Ocsigen documentation on the project website. Take the time to read the tutorials before starting.

- Each Creet must be commanded by a `Lwt` thread.
- It will be displayed as a DOM element (a `<div>` or `<img>` for instance).
- Mouse events must be programmed with the `Lwt_js_event` module of `Js_of_ocaml`. You will find examples of this module in tutorials on the `Ocsigen` website.

Advice:

- Keep your browser debugging tools handy;
- Read the `Ocsigen tutorials`.

# Chapter V

# Mandatory part

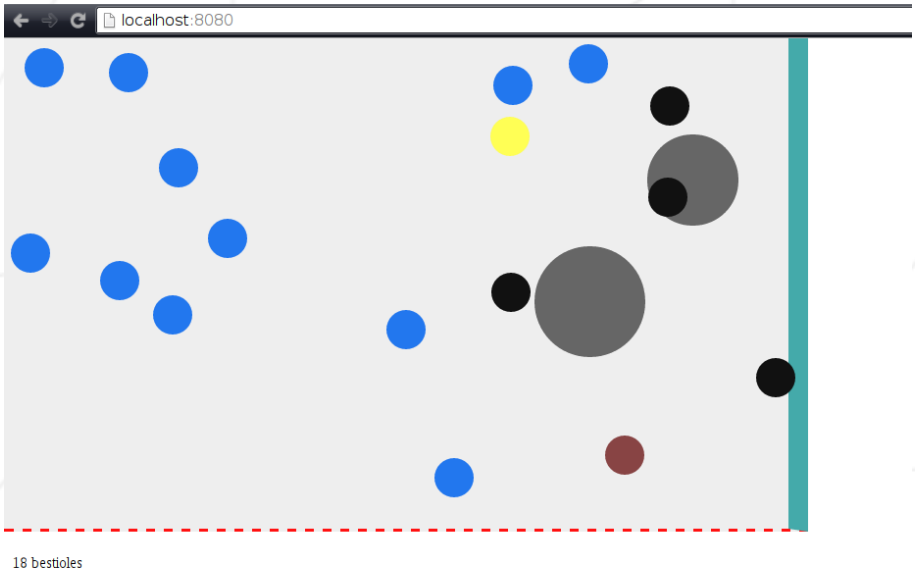
Creets must move randomly on a rectangular space of the browser's window, bordered by a toxic river on top and a hospital where the sick Creets will get healed on the bottom.

Unlucky Creets that will touch the river will get sick. If so, they change color, become contagious and contaminate the ones they touch with a probability.

The goal of the game is to last as long as possible before all the unlucky Creets get contaminated. When this happens, the game is over.

- Creets reproduce spontaneously as long as one healthy Creet is left. As long as they stay healthy, they never die.
- User must be able to grab-move Creets with the mouse to keep them away from the river and take the sick ones to the hospital.
- A sick Creet that randomly touches the hospital won't be healed. Only the ones the user has taken to the hospital will be.
- When a Creet is grabbed by the user, it becomes "invulnerable" and cannot be contaminated if it's healthy.
- A Creet moves in straight lines but might randomly change direction. These unexpected changes must not happen too often to help the player plan the Creets' path, but still create surprise somehow.
- A Creet that touches the edges of the game area rebounds realistically, whichever edge it is.
- A Creet that gets contaminated instantly changes color and gets 15% slower.
- A contaminated Creet that touches a healthy Creet has a 2% risk of contaminating it at each iteration. There is no collision ("no rebound"); Creets walk over each other.
- A Creet that gets sick has a 10% risk of becoming berserk! In that case, the Creet's color changes from the contaminated one, and its diameter slowly grows until it has quadrupled by the end of its life, making it more likely to contaminate others, etc.

- Also, a Creet that gets contaminated has a 10% risk of becoming mean! When this happens, the Creet gets a different color from the contaminated Creets and the Berserks. It shrinks 15% smaller than its regular size but starts running after the other healthy Creets to contaminate them willingly! Nasty little Creet!
- A contaminated Creet cannot be both Berserk and mean.
- Because of the panic, Creets accelerate in time so the difficulty level increases along until the player gets overwhelmed ... and loses. You're free to set the balance as you see fit. The game should be very easy at first and grow harder with time. The player should feel confident in the beginning, until the game turns into a nightmare demanding relentless clicking skills.
- You will choose the number of Creets according to the size you will give them to offer a balanced playability and a clear display. Very small Creets, for instance, would probably make the game a little dull.
- You can adapt the instructed probabilities if and only if you feel like it will help the balance of the game without changing its nature.
- When no living or healthy Creet is left, the game stops with a displayed GAME OVER message.



# Chapter VI

# Bonus part

When you committed yourself to a project and you're happy with the result, it's only natural to want to keep going! Of course, the bonus part will count only if the mandatory part has been thoroughly achieved.

Here is a list of bonus ideas for this project:

- Try to make your game graphically pleasant;
- Implement a way to control the simulation parameters with various forms, cursors or checkboxes on the page;
- Optimize the collision detection between the Creets using a quadtree for instance.

The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

# Chapter VII

# Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.