

Synchronous Retrogames in HTML5

Retrogames were played in the dark arcade rooms of the past millenium. Such games were written in assembly as to squeeze every bit of performance of hardware that wouldn't go past a few Megahertz. Nowadays, a mere smartphone harnesses several Megabytes of RAM and a multiple processor cores clocked at several Gigahertz each. Such power allows us to revisit game programming: ease of development takes the lead while performance is sidelined.

Web browsers are ubiquitous - and are quickly becoming the defacto platform to run multiplatform apps. Furthermore, the new web standard HTML5[1] greatly simplifies the deployment of dynamic application through the Javascript language. Notably, the `canvas`[2] interface allows to draw objects, animate them, and intercept inputs from mouse or keyboards in a Web page.

The canvas API imposes a reactive programming paradigm: in order to update the canvas, one should register a callback function through `requestAnimationFrame()`. [3] One has to do the same to process input.

The reactive limitations imposed by the canvas API is similar to what is used in control-command systems. Several languages, based on the synchronous dataflow paradigm, have been created in order to ease development of such systems. A program written in such a language processes a flow of events - player inputs - and produces a flow of actions corresponding to instructions that will be transmitted to the actuators.

In this internship, we aim at applying this programming model to the implementation of gameplay code. We will focus on the application of the synchronous dataflow formalism to retrogame programming:

- We describe a first approach implementing a Snake[4] clone in pure javascript.
- We design a synchronous dataflow language integrating the HTML5 canvas API.
- We devise a compiler compiling from this language to Javascript, The compiler is written in Ocaml, and will be bootstraped through `js_of_ocaml`.
- We validate this second approach by implementing an other retrogame - this time using this brand new language.

Consider the following javascript code, which simply moves a square from the left to the right:

```
var x = 0;
var y = 150;

var dir_enum = Object.freeze({
  LEFT: 37,
  RIGHT: 39,
});
```

```

dir = 0;

document.addEventListener("keypress", keyPressHandler, false);

function keyPressHandler(e) {
  dir = e.keyCode;
  switch (dir) {
    case dir_enum.RIGHT:
      x += 20;
      break;
    case dir_enum.LEFT:
      x -= 20;
      break;
    default:
      break;
  }
}

function draw() {
  requestAnimationFrame(draw);

```

The callback registration is cumbersome. Furthermore, one has to use many global variables, which lead to confusing code.

Now compare it to the much nicer *sdf* code:

```

#static diff

type Dir = Up | Down

node move_x (key: Dir, x: int) return (x': int)
let
  x' = match key with
    | Up -> x - diff
    | Down -> x + diff
  end
tel

node main (key : Dir) return (x : int)
let
  x = 0 fby move_x(key, x)
tel;

```

Everything related to event programming is abstracted in the structure of the language.

Such code is then compiled to the following javascript:

```

var dir_enum = Object.freeze({
  LEFT: 37,
  RIGHT: 39,
});

function main() {
  this.x = 0;
}

main.prototype.step = function (key) {
  var tmp_x = move_x_node.step(key, this.x)
  this.x = tmp_x;
  return tmp_x;
}

function move_x() {}

move_x.prototype.step = function (dir, x) {
  switch (dir) {
    case dir_enum.Left:
      x_1 = x - diff;
      break;
    case dir_enum.Right:
      x_1 = x + diff;
      break;
    default:
  }
  var new_x = x_1;
  return new_x;
}

```

[1] Mozilla Developer Network. Html5 developer guide, November 2016.
<https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>

[2] Mozilla Developer Network. Canvas api, November 2016. <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas>

[3] Mozilla Developer Network. requestAnimationFrame api, November 2016.
<https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame>

[4] Wikipedia. Snake, November 2016. https://en.wikipedia.org/wiki/Snake_%28video_game%29