Systèmes dynamiques

Guillaume Pelletier-Auger

8 avril 2017

Résumé

Définition, exploration et catalogage des systèmes dynamiques.

1 Définition du prototype *System*

Qu'est-ce qui définit un système dynamique comme je les conçois ? Essayons un premier jet.

```
var System = function(system){
    this.pos = system.pos || {x : 0, y : 0, z : 0};
    this.n = system.n || 0;
    this.animation = system.animation || false;
    this.backgroundColor = system.backgroundColor || 0;
    this.setup = system.setup;
    this.iterativeFunction = system.iterativeFunction;
    this.displayFunction = system.displayFunction;
    this.animation = system.animation || false;
};

System.prototype.runIterativeFunction = function() {
    this.pos = this.iterativeFunction(this.pos.x, this.pos.y, this.pos.z, this.constants);
    this.n++;
};

System.prototype.runDisplayFunction = function() {
    this.displayFunction(this.pos, this.n, this.translate, this.scale);
};
```

2 Définition des systèmes individuels

```
var dune001 = new System({
    pos : {x : 0, y : 0, z : 0},
    n : 0,
    constants : {
        a : 5,
        b : 3,
        c : 2,
        d : 1,
        e : 2,
        f : 1
    },
    animation : false,
    iterationsPerFrame : 1500,
    backgroundColor : 0,
    setup : function() {
        frameRate(30);
        background(0);
        fill(255, 5);
        noStroke();
    },
    iterativeFunction : function(x, y, z, c) {
        //c is a set of constants.
        var v = {
            x : sin(c.a * x) + cos(c.b * y) - tan(c.c * z),
            y : cos(c.d * y) + cos(c.e * x) + tan(c.f * z),
            z : z + 0.1
        };
        return v;
    },
    displayFunction : function(v, n) {
        var red = map(abs(sin(n / 10)), 0, 1, 255, 100);
        var green = map(abs(cos(n / 30)), 0, 1, 200, 40);
        var blue = map(abs(sin(n / 10)), 0, 1, 0, 255);
        blendMode(ADD);
        fill(red, green, blue, 15);
        ellipse(v.x * 100, v.y * 100, 0.5);
    }
});
```

Les systèmes devraient-ils également avoir des valeurs *scale* et *translate* ? Ça me permettrait de réutiliser les mêmes fonctions d'affichage mais de modifier la zone affichée. D'expérience, c'est quelque chose que je voudrai faire très souvent.

3 Définition de fonctions indépendantes

Les fonctions itératives peuvent être définies indépendamment des systèmes dynamiques, et utilisées par ceux-ci. Même chose pour les fonctions d'affichage.

```
var duneFunction = function(x, y, z, c) {
    var v = {
        x : sin(c.a * x) + cos(c.b * y) - tan(c.c * z),
        y : cos(c.d * y) + cos(c.e * x) + tan(c.f * z),
        z : z + 0.1
    };
    return v;
};

var colorFunction001 = function(v, n, translate, scale) {
    var red = map(abs(sin(n / 10)), 0, 1, 255, 100);
    var green = map(abs(cos(n / 30)), 0, 1, 200, 40);
    var blue = map(abs(sin(n / 10)), 0, 1, 0, 255);
    blendMode(ADD);
    fill(red, green, blue, 15);
    ellipse(translate.x + v.x * scale.x, translate.y + v.y * scale.y, 0.5);
};
```

4 Une possible fonction pour dupliquer un système

Devrais-je me créer une fonction qui ne servirait qu'à dupliquer un système entier, pour ensuite le modifier ? Ça pourrait ressembler à ça :

```
var dune003 = dune002.clone();
dune003.scale = {x : 200, y : 0};
```

Et la fonction copiante ressemblerait à ça :

```
System.prototype.clone = function() {
    return {
        pos : {x : this.pos.x,
            y : this.pos.y,
            z : this.pos.z},
        n : this.n,
        constants : this.constants,
        animation : this.animation,
        iterationsPerFrame : this.iterationsPerFrame,
        backgroundColor : this.backgroundColor,
        setup : this.setup,
        iterativeFunction : this.iterativeFunction,
        displayFunction : this.displayFunction
    };
};
```

En fait, peut-être qu'un système pourrait tout simplement cloné (et modifié) comme ça :

```
var dune003 = new System(dune002);
dune003.scale = {x : 200, y : 0};
```

5 Affichage des systèmes dynamiques avec une matrice de densité

Si je veux vraiment bien faire les choses, il me faut développer un système entièrement différent pour l'affichage de mes systèmes dynamiques. C'est-à-dire que je dois me servir de ses systèmes pour remplir une grande matrice avec des informations de densité, et ensuite afficher ces densités en les visualisant.

Cette technique, beaucoup plus compliquée, me permettra de nombreuses choses. Premièrement, elle devrait être beaucoup plus rapide. Deuxièmement, elle me permettra d'afficher mes images en ajustant les courbes de couleur comme dans Photoshop, mais dans mon sketch p5.js lui-même. J'obtiendrai donc un contrôle beaucoup plus grand de la qualité de mes images. Et troisièmement, c'est avec cette technique je devrais pouvoir mélanger p5.js et Node.js et exporter d'immenses images directement sur mon disque dur, sans passer un fureteur. Ces trois bénéfices ont clairement été mentionnés ci-dessus par ordre de complexité.

Je dois définir mes systèmes de façon à ce qu'ils puissent être utilisés par p5.js et par Node.js. Lorsque j'exporterai des images dans Node.js, en fait, je n'ai absolument pas besoin de passer par p5.js.

5.1 Travail initial pour comprendre l'array *pixels* dans p5.js

```
function setup() {
  createCanvas(100, 100);
  background(0);
  fill(255);
  noLoop();
  noStroke();
}

function draw() {
  translate(width / 2, height / 2);
  ellipse(50,0,5);
  loadPixels();
  console.log(pixels.length);
  console.log(pixels[0]);
  //This array is the size
  //(include an appropriate factor for pixelDensity)
  //of the display window x4, representing the R, G, B, A
  console.log(200*200*4);
  console.log(2560*2*1600*2*4);
}
```