Details of Catodo's work for Lampo art exhibition

The work of Catodo is a generative art project based on the architecture of the Aurum building in Pescara, the structure that will accomodate the Lampo exhibition on new frontiers of digital art.

The architecture of Aurum in Pescara has almost a circular shape that remember the structure of an amphitheater (Figure 1).

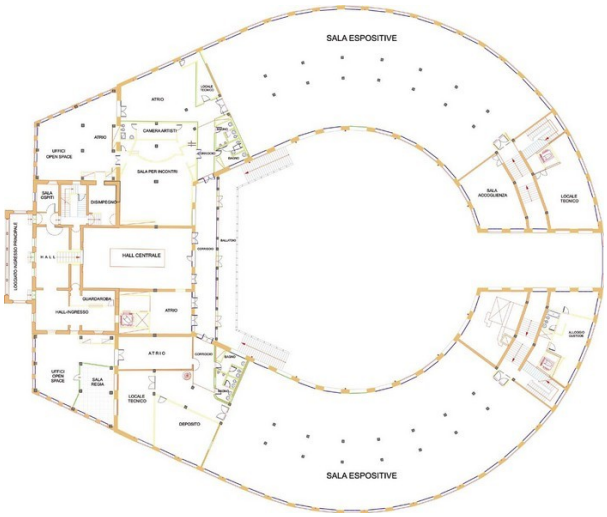

Figure 1 – Planimetry of the Aurum

Inspired by this shape, i built a computer algorithm that:

1. generates a variable number of points on a circle;
2. moves these points by a random value along the x and y axes;
3. joins these points using a *spline* based on the algorithm of *Catmull-Rom*.

A schema of this algorithm is reported in figure 2.

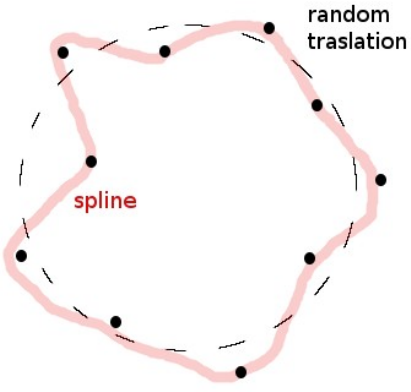

Figure 2 – *Random points joined by a spline*

The result of this algorithm has been used as pattern for the generation of the images. I used a random approach in the choise of the position, of the dimension of the radius, of the size of the lines, and for the value of the alpha color that has determined the opacity of the images.

The management of the opacity has created an interesting effect of blurring that has produced a visual result similar to a complex system.

I experimented with three different predominant colors (green, red, and blue) and with a neutral color, white, generating three images in high resolution (100x70 cm) that represent the interconnection of two complex networks.

The generated images are represented by the Iteration #5132, #2149, #6534 (Figure 3).

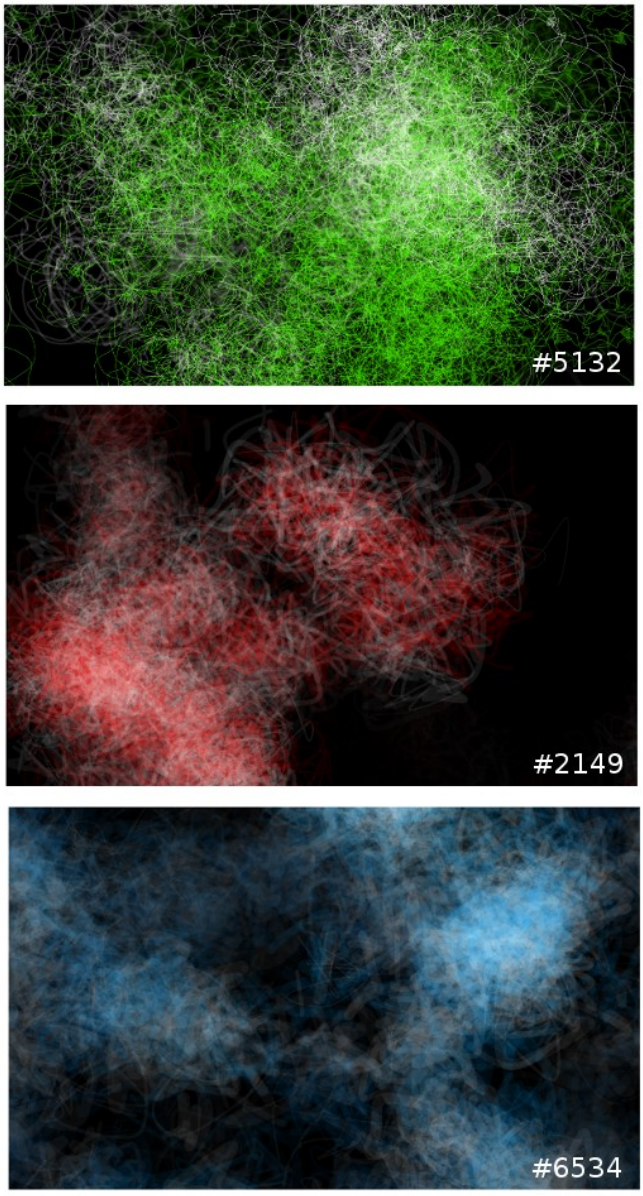

Figure 3: final interactions

Note about the author

Enrico "Catodo" Zimuel (Pescara, 1974) is a creative coder who started to program with a home computer Texas Instruments TI99/4A and never stopped. He released works on electronic and experimental music with the group Gli Elettrodi and with the Unit project, together with the artist Globster, at the end of '90. He published these works with the Kutmusic label. He worked with fractal and generative art using different systems, like the home computer Commodore 128. He writes articles about eletronic art on the website aboutart.it. He is interested in net art and generative art using open source systems. He holds a B.Sc. (Hons) in Computer Science and Economics from the University "G.D'Annunzio" of Pescara and he did computer science research at the Informatics Institute of the University of Amsterdam. He studied algorithms based on cellular automata with Stephen Wolfram at the Brown University (USA). He works and lives in Turin (Italy).

Technical Appendix

In this Appendix, I reported the source code of the algorithms that I used to generate the previous images.

Iteration #5132

Image generated in 42 minutes using an Intel Core i5 a 3.3 Ghz with 8 Gb of RAM, with the operating system Ubuntu Linux 12.04 and Processing language.

```
float xc, yc, r, g, b;
float num = 1;
int[][] colors = {{30,232,0},{150,150,150}};

void setup() {
  size(3380, 2048);
  colorMode(RGB, 100);
  background(0, 0, 0);
  smooth();
  noFill();
  strokeCap(ROUND);
  strokeJoin(ROUND);
  xc = width/2 + random(-width/4, width/4);
  yc = height/2 + random(-height/4, height/4);
}

void draw() {
  xc += random(-width/20,width/20);
  yc += random(-width/20,width/20);
  int c = round(num) % 2;
  stroke(colors[c][0],colors[c][1],colors[c][2],num);
  strokeWeight(1/num * 150);
  generate(xc+noise(xc), yc+noise(yc), round(num % 34));
  num += 0.2;
  if (num>80) {
    num = 0;
    xc = width/2 + random(-width/4, width/4);
    yc = height/2 + random(-height/4, height/4);
  }
}

void generate(float xc, float yc, int num_points){
  float x;
  float y;
  beginShape();
  float al = 2*PI / num_points;
  float rd = random(-width/15,width/15);
  beginShape();
  for (int i = 0; i < num_points+3; i++)
  {
    x = xc + cos(al*i) * rd + random(-30, 30);
    y = yc + sin(al*i) * rd + random(-30, 30);
    curveVertex(x, y);
  }
  endShape();
}
```

Iteration #2149

Image generated in 19 minutes using an Intel Core i5 a 3.3 Ghz with 8 Gb of RAM, with the operating system Ubuntu Linux 12.04 and Processing language.

```
float xc, yc, r, g, b;
float num = 1;
int count = 1;
int[][] colors = {{125,0,0},{159,159,159}};

void setup() {
  size(3380, 2048);
  colorMode(RGB, 100);
  background(0, 0, 0);
  smooth();
  noFill();
  strokeCap(ROUND);
  strokeJoin(ROUND);
  xc = width/2 + random(-width/4, width/4);
  yc = height/2 + random(-height/4, height/4);
}

void draw() {
  xc += random(-width/20,width/20);
  yc += random(-width/20,width/20);
  int c = round(num) % 2;
  stroke(colors[c][0], colors[c][1], colors[c][2], count/100);
  strokeWeight(random(width/100));
  generate(xc+noise(xc), yc+noise(yc), round(num));
  translate(width/2, height/2);
  rotate(PI/3.0*random(num));
  num += 0.2;
  count++;
  if (num > 50) {
    num = 0;
    xc = width/2 + random(-width/4, width/4);
    yc = height/2 + random(-height/4, height/4);
  }
}

void generate(float xc, float yc, int num_points){
  float x;
  float y;
  beginShape();
  float al = 2*PI / num_points;
  float rd = random(-width/15,width/15);
  beginShape();
  for (int i = 0; i < num_points+3; i++)
  {
    x = xc + cos(al*i) * rd + random(-width/20, width/20);
    y = yc + sin(al*i) * rd + random(-width/20, width/20);
    curveVertex(x, y);
  }
  endShape();
}
```

Iteration #6534

Image generated in 31 minutes using an Intel Core i5 a 3.3 Ghz with 8 Gb of RAM, with the operating system Ubuntu Linux 12.04 and Processing language.

```
float xc, yc, r, g, b;
float num = 1;
int count = 1;
int[][] colors = {{0,64,173},{159,159,159}};

void setup() {
  size(3380, 2048);
  colorMode(RGB, 100);
  background(0, 0, 0);
  smooth();
  noFill();
  strokeCap(ROUND);
  strokeJoin(ROUND);
  xc = width/2 + random(-width/4, width/4);
  yc = height/2 + random(-height/4, height/4);
}

void draw() {
  xc += random(-width/20,width/20);
  yc += random(-width/20,width/20);
  int c = round(num) % 2;
  stroke(colors[c][0], colors[c][1], colors[c][2], count/400);
  strokeWeight(random(width/50));
  generate(xc+noise(xc), yc+noise(yc), round(random(num)));
  translate(width/2+random(width/10), height/2+random(height/10));
  rotate(PI/3.0*random(num));
  num += 0.2;
  count++;
  if (num > 50) {
    num = 0;
    xc = width/2 + random(-width/4, width/4);
    yc = height/2 + random(-height/4, height/4);
  }
}

void generate(float xc, float yc, int num_points){
  float x;
  float y;
  beginShape();
  float al = 2*PI / num_points;
  float rd = random(-width/15,width/15);
  beginShape();
  for (int i = 0; i < num_points+3; i++)
  {
    x = xc + cos(al*i) * rd + random(-width/20, width/20);
    y = yc + sin(al*i) * rd + random(-width/20, width/20);
    curveVertex(x, y);
  }
  endShape();
}
```