Scheda delle opere per la mostra Lampo
di *Catodo*

I lavori proposti sono il risultato di un progetto di arte generativa realizzato pensando all'architettura dell'Aurum di Pescara, struttura che ospiterà la mostra Lampo sulle nuove frontiere dell'arte digitale.

La struttura architettonica dell'Aurum di Pescara ha una forma quasi circolare che ricorda un anfiteatro (Figura 1).

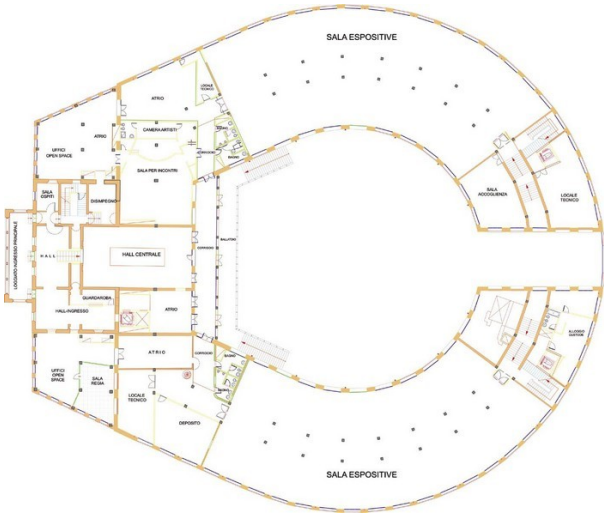

Figura 1 - Planimetria dell'Aurum di Pescara

Ispirandomi a questa forma, ho ideato un algoritmo per computer che:

1. genera un numero variabile di punti su una circonferenza;
2. trasla questi punti di un fattore casuale lungo l'asse delle x e delle y;
3. unisce questi punti tramite una *spline* con il noto algoritmo *Catmull-Rom.*

Uno schema di questo algoritmo è riportato in figura 2.

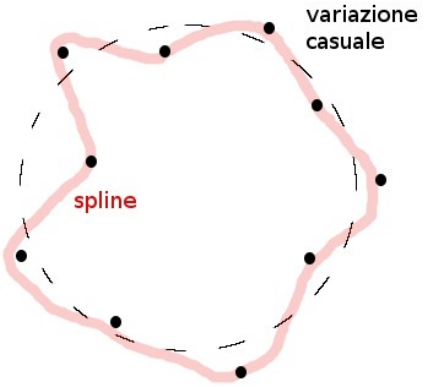

Figura 2 - Punti casuali uniti da una *spline*

Il risultato di questo algoritmo è stato utilizzato come *pattern* per la generazione delle opere visive. E' stato utilizzato un approccio casuale per determinare la posizione, la dimensione del raggio, lo spessore delle linee e l'intensità del canale alfa del colore che ha determinato l'opacità delle immagini. La gestione dell'opacità delle immagini ha generato un effetto interessante di sfocatura che ha prodotto il risultato visivo di un sistema complesso.

Sperimentando con l'utilizzo di una tonalità di colore predominante (verde, rosso e blu) e con un colore neutro, il bianco, sono state generate tre immagini in alta risoluzione (100x70 cm) che rappresentano l'interconnessione di due reti complesse.

Le immagini generate sono rappresentate dalle Iterazioni #5132, #2149, #6534 (Figura 3).

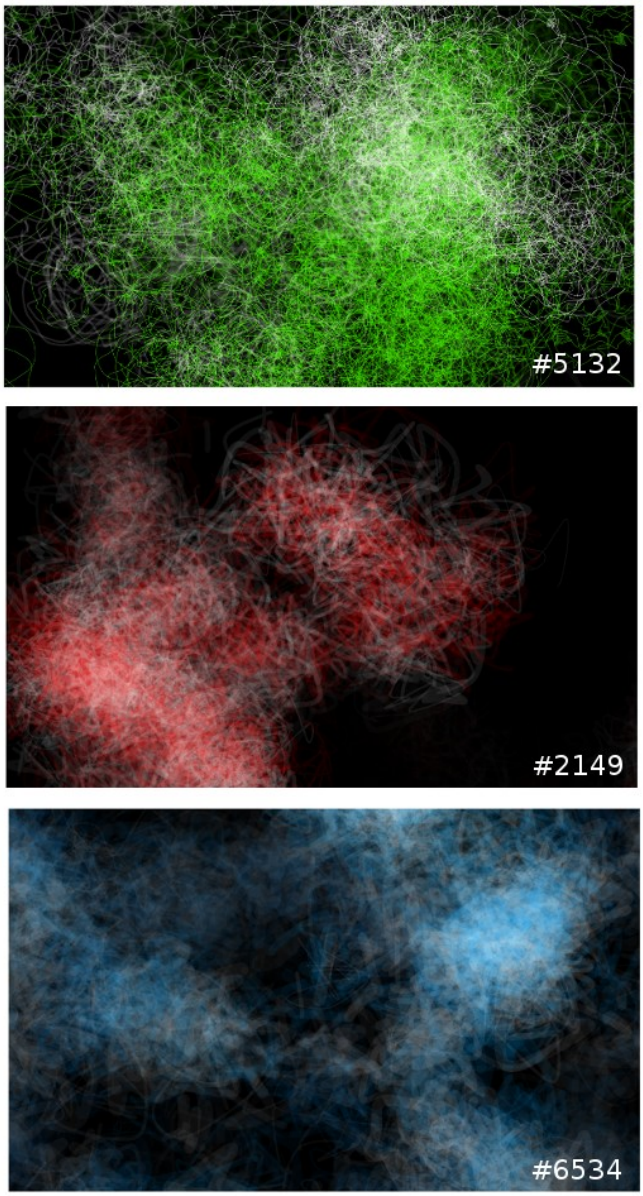

Figura 3: iterazioni finali

Note sull'autore

Enrico "Catodo" Zimuel (Pescara, 1974) è un programmatore creativo che ha iniziato a sviluppare con un Texas Instruments TI99/4A e dall'ora non ha più smesso. Ha realizzato alcuni lavori di musica elettronica e sperimentale con il gruppo Gli Elettrodi ed il progetto Unit, insieme con l'artista Globster alla fine degli anni '90, pubblicando per l'etichetta Kutmusic. Si è poi occupato di arte frattale ed arte generativa realizzando installazioni con diversi sistemi, tra i quali anche home computer come il Commodore 128. Attualmente si occupa di net art ed arte generativa su sistemi *open source*. Scrive di arte elettronica per aboutart.it. Si è laureato in Economia Informatica presso l'Università "G.D'Annunzio" di Pescara ed ha svolto attività di ricerca presso l'Informatics Institute dell'Università di Amsterdam. E' stato un allievo di Stephen Wolfram presso la Brown University (USA) dove si è specializzato sugli algoritmi basati su automi cellulari. Vive e lavora a Torino.

Appendice tecnica

In questa appendice tecnica sono riportati gli algoritmi che sono stati utilizzati per la generazione delle immagini.

Iterazione #5132

Immagine generata in 42 minuti utilizzando un Intel Core i5 a 3.3 Ghz con 8 Gb di RAM, con sistema operativo Ubuntu Linux 12.04 e linguaggio di programmazione Processing.

```
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();
}
```

Iterazione #2149

Immagine generata in 19 minuti utilizzando un Intel Core i5 a 3.3 Ghz con 8 Gb di RAM, con sistema operativo Ubuntu Linux 12.04 e linguaggio di programmazione Processing.

```
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();
}
```

Iterazione #6534

Immagine generata in 31 minuti utilizzando un Intel Core i5 a 3.3 Ghz con 8 Gb di RAM, con sistema operativo Ubuntu Linux 12.04 e linguaggio di programmazione Processing.

```
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();
}
```