```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>
```

código concreto

```
    </title>
    <meta charset="utf-8">
  </head>
  <body>
  </body>
</html>
```

```
<!DOCTYPE html>
<html lang=”pt-br”>
  <head>
    <title>
```

código concreto

```
    </title>
    <meta charset=”utf-8”>
  </head>
  <body>
  </body>
</html>
```

Código Concreto

Tobias Gaede

Trabalho final do unidade curricular
Pensamento da Arte Atual PENAA
Doutoramento em Artes Plásticas
FBAUP

Professores PENAA
Filipa Cruz
Samuel Silva

Orientação
André Rangel

Junho de 2024
Porto, Portugal

ISBN
978-65-01-18816-4

Fonte
Courier New

ao
ChatGPT

Código Concreto

Isso não é um manifesto. Nem é uma tentativa de seguir à risca o que delinearam o *Manifesto Concretista* ou o *Plano-Piloto para a Poesia Concreta*[1]. Já se vão quase 70 anos desde que Haroldo de Campos, Augusto de Campos e Décio Pignatari os publicaram, e não é agora que se vai reafirmá-los ou rechaçá-los, como fizeram os neoconcretos frente às discordâncias que os motivaram a se reinventarem.

Nem quero, também, pleitear qualquer tipo de primazia quanto à relação dos códigos de programação e da poesia, ou com o concretismo… Waldemar Cordeiro, em sua *arteônica*, já rondava esse campo[2]; a Poesia Semiótica também já codificava — embora por outros critérios — seus textos ideogramáticos; Brion Gysin já experimentava, na década de 1950, a criação de textos pelo método da "poesia permutacional"[3]; mais recentemente, propostas baseadas na

[1] Campos, A. (1965). Teoria da poesia concreta. Edições Invenção.

[2] https://www.waldemarcordeiro.com/arteonica (acesso em junho de 2024)

[3] Reis, P. (2011). Primórdios da poesia em computador - anos 60, 70 e 80 do século XX. Romance Notes 51(3), 293-304. https://doi.org/10.1353/rmc.2011.0036.

escrita de algoritmos, como as de Zackary Scholl[4], são recorrentes; a estética dos códigos é presente em trabalhos poéticos como os de Soraya Roberta[5].

O que quero fazer aqui é tornar o código concreto, literalmente: que assuma um corpo tangível, vire papel. E torná-lo literatura, também literalmente: um código que possa ser tanto lido por pessoas como decodificado por máquinas.

*

Mesmo com todas as interferências poéticas que o texto de programação de ***Código Concreto*** sofreu, ele continua sendo funcional, respeitando as regras sintáticas e semânticas da linguagem de programação e da biblioteca de *JavaScript* usada aqui — o *p5.js* —, gerando assim um resultado visual por meio de seu processamento. Cada poema de ***Código Concreto*** é perfeitamente legível para a máquina. Isso está demonstrado nos *links* em *QRcode* que acompanham cada uma das experiências aqui reunidas. Basta acessá-los para ver a versão não-concreta do trabalho.

Quanto ao leitor humano, ele poderá encontrar palavras bem reconhecíveis em meio às *tags*, métricas entre as equações, ritmo entre os algoritmos. Uma parte desse texto é composta por comentários (indicados por "//") que não têm qualquer função no desempenho do código. Outra parte é usada como variável (elemento antecedido por "*let*" e repetido ao longo do bloco de código), ao qual é atribuída função determinante no resultado final no processamento do projeto. Ao longo da composição, esses termos passam a se combinar, produzindo significados à medida que

[4] https://www.vice.com/pt/article/53y7wd/o-poema-que-passou-no-teste-de-turing (acesso em junho de 2024)

[5] https://poesiacompilada.com/sobre.html (acesso em junho de 2024)

se reúnem. Nem sempre isso implica na construção de sentenças completas, aproximando-se talvez daquela "totalidade sensível 'verbivocovisual'"[6] da qual falavam os poetas concretos.

Divido o livro em duas partes. Na primeira estão aqueles poemas de temática relacionada ao ato de criar, à gênese, aos mitos fundadores. A criação em questão é a artística, mas também é a de criaturas. A segunda seção tem conteúdos com mais carga ética e moral, talvez de costumes. Acredito que, dessa forma, possa se dar certa ordem lógica à publicação.

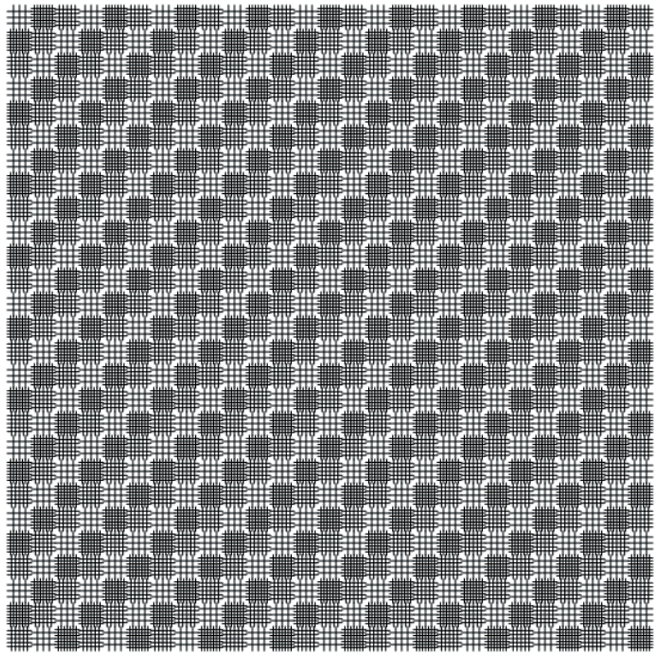
Resta ainda esclarecer que eu sou o responsável pelo que é poético nesses textos. Já a escrita dos códigos é, em grande parte, de responsabilidade de um outro. Não é um auxiliar humano, é um dispositivo artificial, o *ChatGPT*. Explicito o uso desse artifício com certo destaque, pois o uso dessa ferramentas de inteligência artificial conversacional é parte central da investigação que desenvolvo em minha pesquisa de doutoramento e elemento importante no método de criação artística que experimento. Portanto, ao final desta publicação, dou acesso aos processos que levaram a esse resultado, do qual agora o prezado leitor (humano ou não) poderá desfrutar.

Tobias Gaede

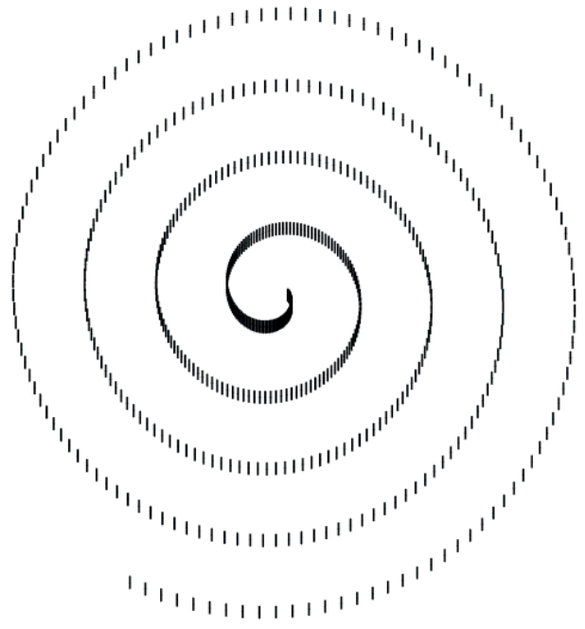
Porto, 26 de junho de 2024

[6] Trecho de "Manifesto Concretista: poesia concreta: um manifesto".

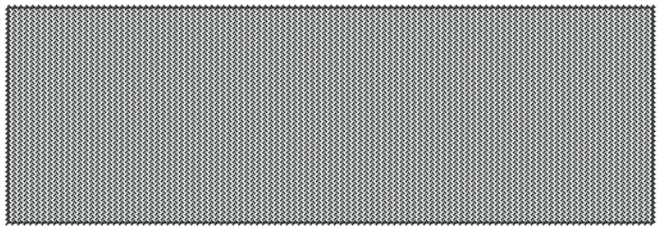
```
function setup() { // e no princípio, era o Código
createCanvas(1080,1530); // tudo foi feito por ele
noLoop();} function draw() { background(255); // é
// e nada do que é pode ser sem que antes ele seja
let fogo = (width - 540) / 2 ; // algo resplandece
// ainda distante prenucia a chegada de boas novas
let vento =(height - 540)/2; let água = 20; // vem
for ( let x = fogo ; x < fogo + 540 ; x += água) {
for ( let y = vento; y < vento + 540; y += água) {
let terra = (x - fogo + y - vento ) %3; // e traça
demiurgo ( x, y, água, terra ) ; } // os contornos
} }function demiurgo(x, y, size, terra){stroke(0);
noFill (); // as trevas não mais hão de prevalecer
let cria; if (terra===0) { cria = 3; // do abismal
} else if (terra===1) { cria=5 ; // aos altos céus
} else { cria = 7 ; } // em cada pixel, cada forma
for (let i=1 ; i<= cria ; i++){ // reúne a criação
let organiza = size /(cria+1); // em pares, nações
line (x,y+i * organiza, x+size, y+i * organiza); }
// seres sem sopro de vida, sem humores, sem carne
for (let i=1; i<=cria; i++) { // ordena cada forma
let organiza = size/(cria+1); // em lugar oportuno
// e permaneçam tal como são desde logo concebidas
line (x+i * organiza, y, x+i * organiza, y+size) ;
} // uns e zeros, presenças e ausências, sim e não
} // código se faz papel, materializa-se entre nós
```



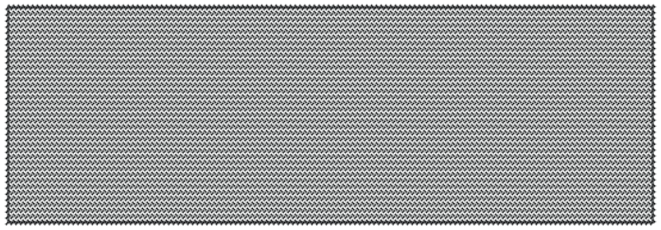
```
                    function
              setup()      {      // em
          createCanvas         (1080,1530)
        ;                         // expansão
      noLoop();}              function draw(){
     background(255)                              ;
   let caos                        =        width/2;
  let silêncio                       =      height/2;
 let profundeza = 10                                 ;
let nada = 0                                         ;
let abismo = PI/60                                   ;
let vazio = 0                                        ;
while (profundeza < 540 / 2)                        {
 let x = caos + profundeza * cos(nada)              ;
  let y = silêncio + profundeza * sin(nada)     ;
   fill(vazio % 256);      ellipse(x,y,0.5,10)   ;
     nada += abismo                             ;
       profundeza += 0.5                      ;
         vazio += 2                          ;
            //                          em
                //            retração
                        }     }
```



```
// urano ouranós  uranus ouranos  urano ouranós  uranus ouranos
function setup () { createCanvas(1080,1530); } function draw ()
// ouranós  uranus ouranos  urano ouranós  uranus ouranos urano
{ background(255); let sémenPaterno=540; let sanguePaterno=180;
// uranus ouranos  urano ouranós  uranus ouranos  urano ouranós
let jorra = (width-sémenPaterno)/2; let ceifa = (height-540)/2;
// ouranos  urano ouranós  uranus ouranos  urano ouranós uranus
stroke (0) ;  rect (jorra, ceifa, sémenPaterno, sanguePaterno);
// ouranos urano  ouranós uranus  ouranos urano  ouranós uranus
rect(jorra,ceifa+540-sanguePaterno,sémenPaterno,sanguePaterno);
// urano  ouranós uranus  ouranos urano  ouranós uranus ouranos
```



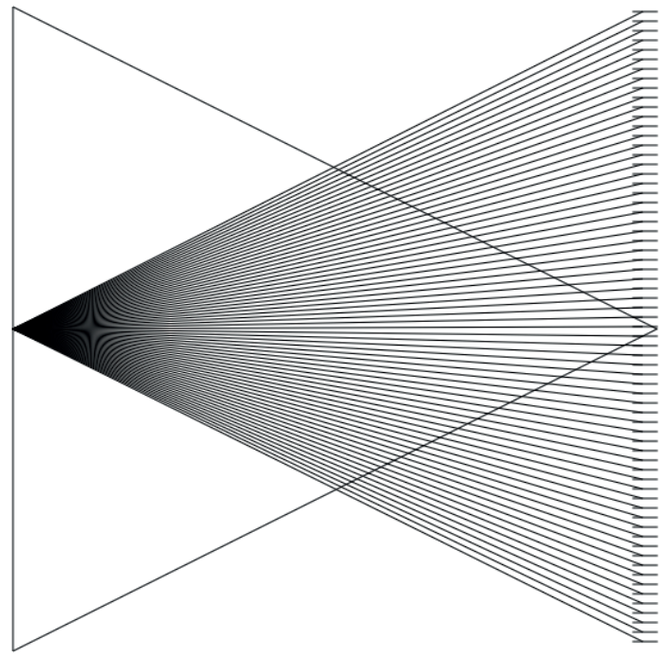
```
for ( let y = ceifa ;  y <= ceifa + sanguePaterno ; y += 4 )  {
// gaia gaía gea  geia gé gaia  gaía gea geia  gé gaia gaía gea
for ( let x = jorra ;  x <= jorra + sémenPaterno ;  x += 5 )  {
// gaía gea  geia gé gaia  gaía gea geia  gé gaia gaía gea geia
line (x, y, x - 2, y + 2) ;  line (x + 2, y, x, y -2) ;   }   }
// gea  geia gé  gaia  gaía gea geia  gé gaia gaía  gea geia gé
for (let y = ceifa +540-sanguePaterno; y<=ceifa+540; y += 5)  {
// geia gé gaia  gaía gea geia  gé gaia gaía  gea geia gé  gaia
for (let x = jorra ;   x  <=  jorra + sémenPaterno ; x += 4)  {
// gé gaia  gaía gea geia  gé gaia gaía  gea geia gé  gaia gaía
line (x, y, x - 2, y + 2) ; line (x + 2, y, x, y -2);   }  }  }
```



II

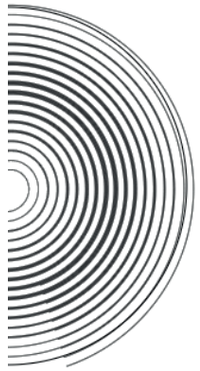
```
function setup () { createCanvas (1080, 1530); background (255) ; // qual será a orientação daquilo que
let sejaLeal=1080;let sejaSão=(width-sejaLeal)/2;let sejaBom=(height-40)/2; for(let i=0;i<15;i++){ // é
line (sejaSão, sejaBom + i*3, sejaSão + sejaLeal, sejaBom + i*3); } } function draw() {} // horizontal?
```

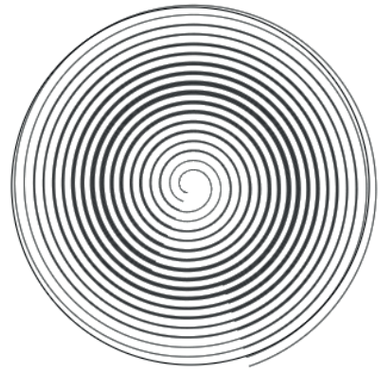
```
//
//    se
//     seguir
function setup() {
createCanvas(1080,1530);
noLoop(); } //      em frente
function draw() { background(255);
let ir = 540; let vem = (width + ir)/2;
let vai = (height-ir)/2;  //  pode-se chegar
scale(-1,1);translate(-width,0); stroke(0); // ao
noFill();beginShape();vertex(vem-ir,vai+ir/2); // ápice
vertex(vem,vai);vertex(vem,vai+ir);endShape(CLOSE);
for(let y  =vai+4; y < vai+ir; y+=8) { //  e
let x1=vem-ir+0; let x2=vem-520;  // ver
line(x1,y,x2,y);x1+=8;x2-=8; // que
if(y<vai+ir/2){ line // resta
(vem,vai+ir/2,x2,y);}{line
(vem,vai+ir/2,x2,y);
//     apenas
// voltar
}}}
```



```
function
{createCanvas(1080,1530);
function draw(){background(255);
let deixa = 100; let vir = width/2;
/2; let será =0; let amanhã=height/2;
function depoisdeamanhã(x, y, r){
for(let i=0.1;i<10;i++){beginShape();
novo < TWO_PI*5.2; novo += 0.001){
/(TWO_PI*6);let sol =x+ dia *cos(novo);
* sin(novo); vertex(sol, luar); }
noFill(); ellipse ( vir, enfim,
depoisdeamanhã(vir,enfim,deixa);
amanhã, deixa*3, deixa*3);
(será, amanhã, deixa);
}
```

```
setup()
noLoop(); }
// o que virá?
let enfim = height
// diferente do que
// sempre conheci?
for (let novo = 2;
let dia=r*(novo+i*3)
let luar = y + dia
endShape() ; } }
deixa*3,deixa*3);
ellipse (será,
depoisdeamanhã
// novamente
// será
```





Histórico de versões do processo de criação
dos códigos de programação no p5.js Web Editor:

Versões finais dos códigos
no p5.js Web Editor:

Audios com a
declamação dos poemas