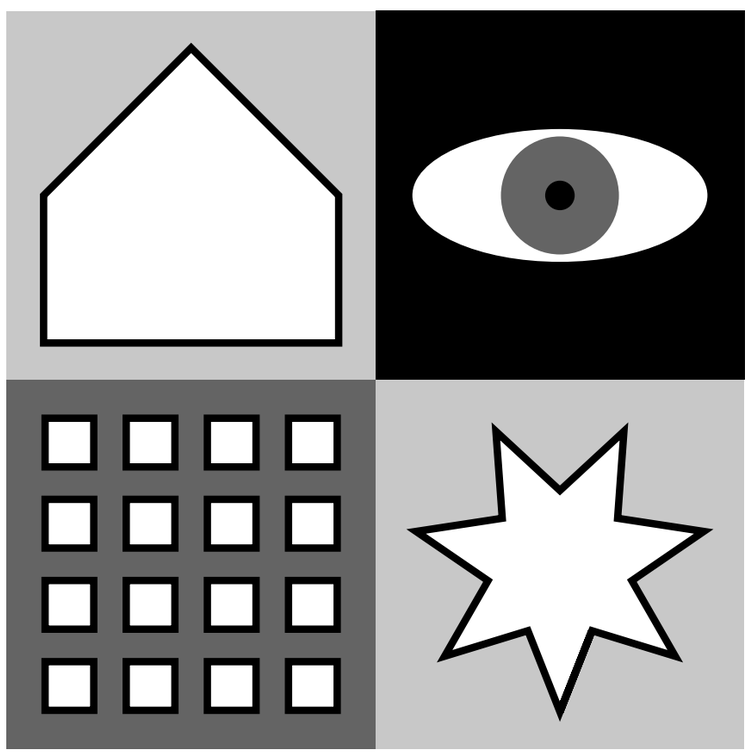
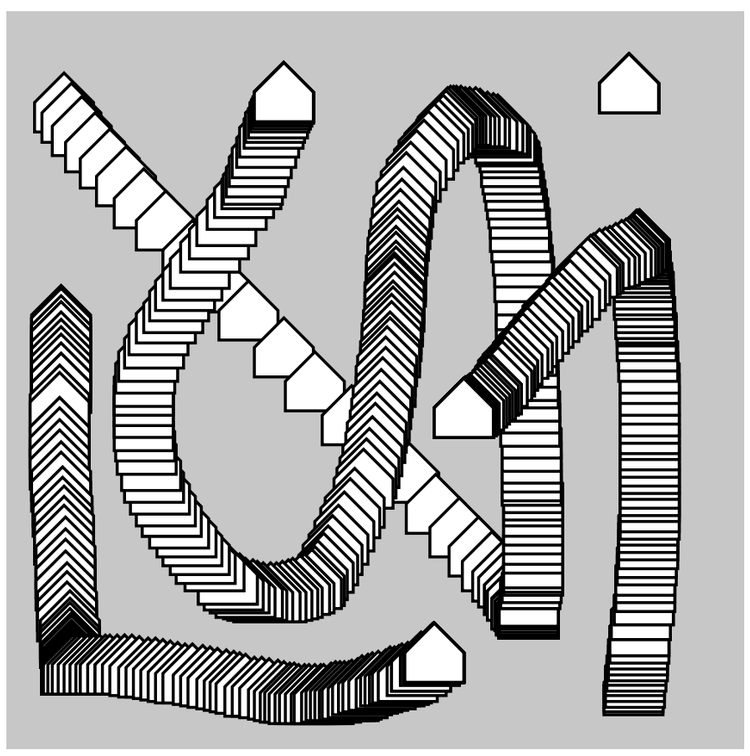
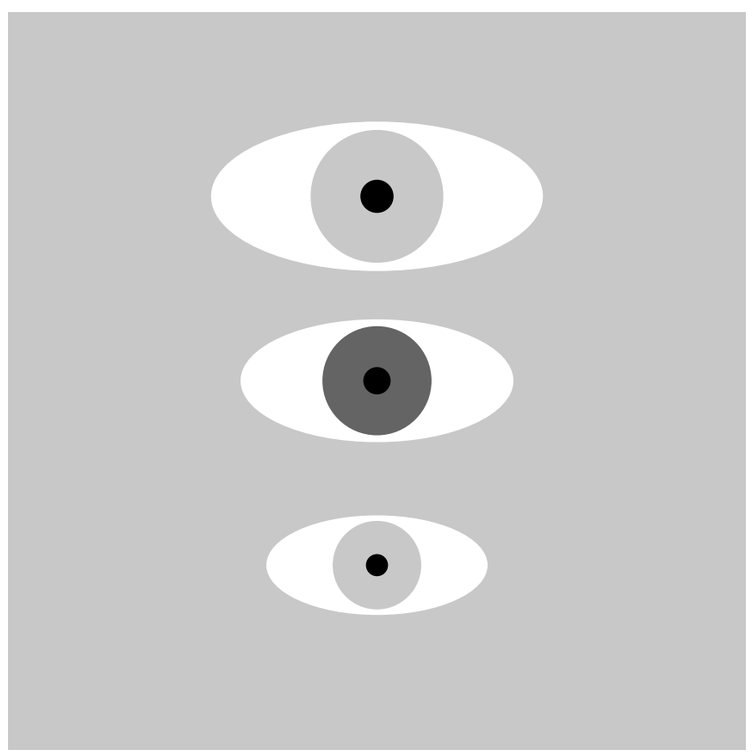
desenho()

```
"""
Exemplo de desenho interativo com setup() e draw().
"""

def setup(): # função executada uma vez no começo
    size(500, 500) # define área de desenho
    background(100) # limpa a área com fundo cinza escuro
    strokeWeight() # ajusta espessura do traço

def draw(): # função executada cerca de 60 vezes por segundo
    if mousePressed: # se o mouse estiver pressionado
        casinha(mouseX, mouseY, 40) # desenhe no x,y do mouse

def casinha(x, y, tamanho):
    """
    Desenhe casinha em x, y com largura e altura 'tamanho'.
    """
    metade = tamanho / 2
    pushMatrix()  # preserva o sistema de coordenadas atual
    translate(x, y)  # translada a origem das coordenadas
    beginShape()  # começa a desenhar a forma/polígono
    vertex(0, -metade)
    vertex(-metade, 0)
    vertex(-metade, metade)
    vertex(metade, metade)
    vertex(metade, 0)
    endShape(CLOSE)  # encerra fechando no 1o vértice
    popMatrix() # retorna o sistema de coordenadas anterior
```

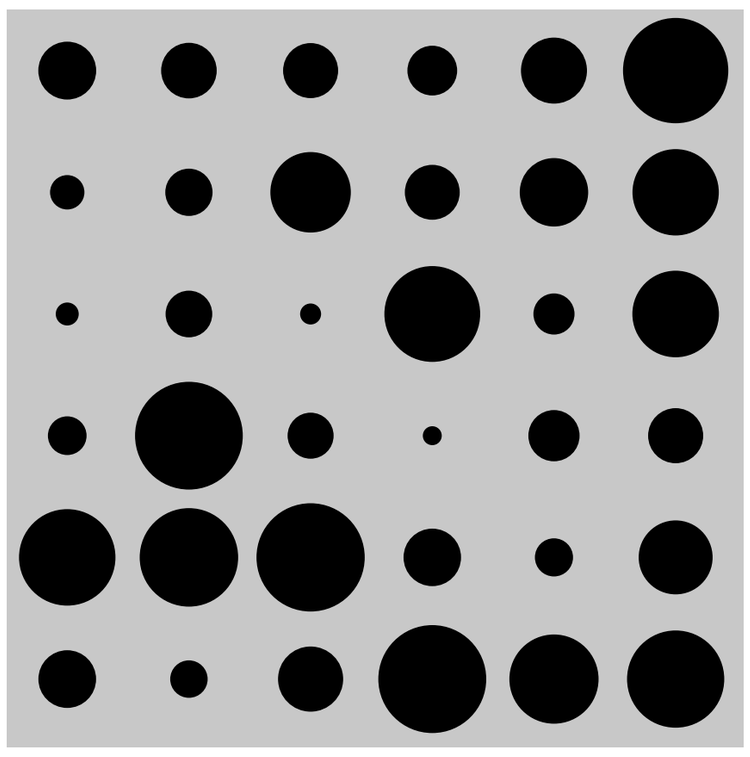




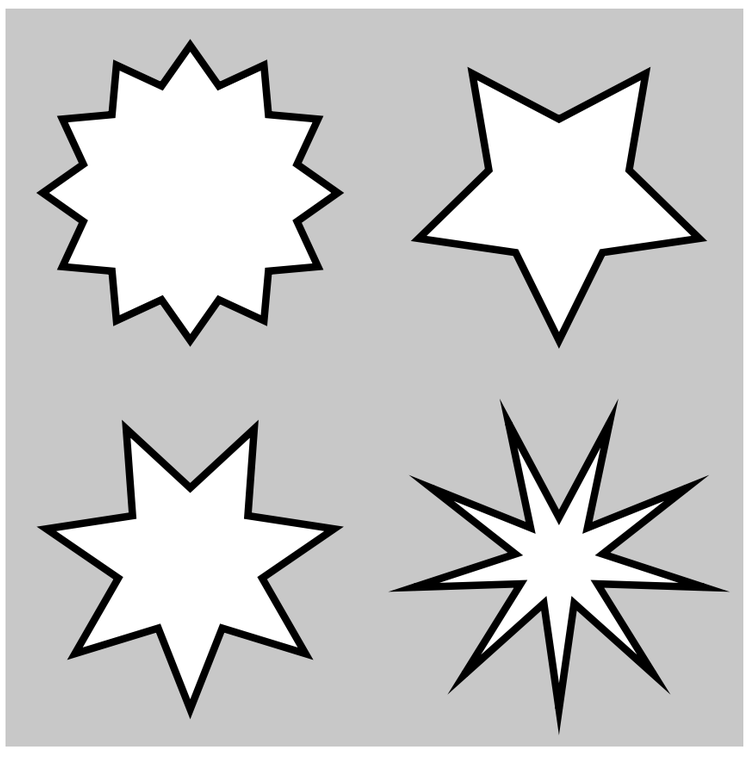
```
"""
Uma função que desenha olhos. Exportação de PDF.
"""
add_library('pdf')  # Inclui biblioteca de PDF do Processing

def setup():
    size(500, 500)
    beginRecord(PDF, 'olho.pdf')
    cinza_escuro = color(100)  # cria um valor de cor
    cinza_claro = color(200)  # cria outro valor de cor
    background(cinza_claro)  # fundo
    olho(width / 2, width / 4, width * .45, cinza_claro)
    olho(width / 2, width / 2, width * .37, cinza_escuro)
    olho(width / 2, width * .75, width * .3, cinza_claro)
    endRecord()  # finaliza salvamento do PDF

def olho(x, y, largura, cor):
    """
    Desenhe olho na posição x, y com largura e cor.
    """
    pushStyle() # preserva os atributos gráficos atuais
    noStroke()  # desenhar formas sem traço de contorno
    fill(255) # preenchimento branco
    ellipse(x, y, largura, largura * .45) # desenha branco
    fill(cor) # cor de preenchimento do parâmetro
    circle(x, y, largura * .4) # desenha iris
    fill(0) # preenchimento preto
    circle(x, y, largura * .1) # desenha pupila
    popStyle() # retorna aos atributos gráficos anteriores
```



```
"""
Uma grade com laços encaixados e elementos variados.
"""

def setup():
    size(500, 500)
    noStroke()  # desenhar formas sem traço de contorno
    fill(0)  # preenchimento preto
    noLoop()  # faz o draw() parar depois de um frame

def draw():
    background(200)  # fundo cinza (e limpa o frame)
    grade(250, 250, 8, width, rnd_circ=True)

def grade(x_centro, y_centro, n, tam_total, rnd_circ=False):
    tam = tam_total / n
    desloc = (tam - tam_total) / 2.
    for i in range(n):
        x = x_centro + desloc + tam * i
        for j in range(n):
            y =  y_centro + desloc + tam * j
            if rnd_circ:
                # no Processing random(inicio, final_ni)
                circle(x, y, random(tam * .1, tam * .9))
            else:
                square(x, y, tam * .75)

def keyPressed():  # no evento de uma tecla ser pressioanada
    saveFrame('grade.png')  # salva uma imagem PNG
    redraw()  # dispara uma nova execução do draw()
```



```
"""
Desenhando quatro estrelas diferentes
"""

def setup():
    size(500, 500)   # área de desenho
    strokeWeight(5)  # espessura do traço
    estrela(125, 125, 12, 100, 75) # estrela de 12 pontas
    estrela(375, 125, 5, 100, 50)  # estrela de 5 pontas
    estrela(125, 375, 7, 100, 50)  # estrela de 7 pontas
    estrela(375, 375, 9, 100, 30)  # estrela de 9 pontas

def estrela(x_centro, y_centro, num_pontas, raio_a, raio_b):
    """
    Desenhe uma estrela em x_centro, y_centro com num_pontas
    raio_a e raio_b são raios dos pontos internos e pontas.
    """
    n = num_pontas * 2 # pontos totais são dobro de pontas
    ang = radians(360. / n) # 360 graus / n em radianos
    beginShape() # começa a desenhar a forma
    for i in range(n):
        r = raio_a if i % 2 == 0 else raio_b
        x = x_centro + sin(ang * i) * r
        y = y_centro + cos(ang * i) * r
        vertex(x, y)
    endShape(CLOSE) # encerra fechando no primeiro ponto
```



#0_out_2019
desenho()

```
"""
Para executar o código instale Processing Modo Python.
Saiba mais em https://desenho.lugaralgum.com

Contribuiram para a impressão inicial:
Adri Patamoma, Advan Shumiski, André Burnier, Bernardo Fontes,
Fábio C. Barrionuevo da Luz, Juan Lopes, Lucia Dossin, Monica Rizzolli,
Otavio Carneiro, Rodolfo Viana, Thais Viana, Uriá Fassina, Yorik van Havre.
(até agora... confira no site!)

Agradecimentos especiais:
Décio Otoni de Almeida, Bernardo Fontes, Monica Rizzolli

Copyright (c) 2019 Alexandre B A Villares
Texto e imagens: CC-BY-SA-NC v4.0 / Código: GPL v3.0
"""

def setup():
    size(500, 500)  # configura a àrea de desenho
    capa()  # depende de funções definidas no folder...

def capa():
    # width é a largura da área de desenho
    metade, quarto = width / 2, width / 4
    noStroke()  # desliga o traço
    fill(100)  # preenchimento cinza escuro
    rect(0, metade, metade, metade)  # fundo para casinha
    fill(0)  # preenchimento preto
    rect(metade, 0, metade, metade)  # fundo preto para o olho
    olho(metade + quarto, quarto, 200)
    fill(255)  # preenchimento branco
    stroke(0)  # traço preto
    strokeWeight(5)  # espessura do traço
    casinha(quarto, quarto, 200)
    grade(quarto, metade + quarto, 4, 220)
    estrela(metade + quarto, metade + quarto, 7, 100, 50)
```

desenho()

```
"""
Poster desenho() #0_outubro_2019 versão 191020
https://desenho.lugaralgum.com
"""

from __future__ import division
from random import choice, seed
from elementos import casinha, estrela, olho
add_library('pdf')

def setup():
    size(1122, 1122)
    noLoop()
    s = 191020
    seed(s)  # define random seed do Python (para choice)
randomSeed(s)  # define random seed do Processing

def draw():
    background(255)
beginRecord(PDF, 'poster-seed-{}.pdf'.format(s))
rectMode(CENTER)
strokeJoin(ROUND)
poster(width / 2., height / 2., 6, width - 100)
endRecord()
```

```
def poster(xo, yo, divisoes, dim_total, elemento=None):
    """
    Faça desenho do poster usando subdivisões recursivas.
    """
    dim = dim_total / divisoes   # dimensão de célula da grade
    offset = (dim - dim_total) / 2
    for i in range(divisoes):
        x = xo + offset + dim * i
        for j in range(divisoes):
            y = yo + offset + dim * j
            sel_elemento = choice((0, 1, 2, 3, 4))
            if elemento is not None:  # elemento grade regular
                desenha_elemento(x, y, dim, elemento)
            elif dim > 20 and random(10) < 7.5:  # subdivisão
                poster(x, y, 3, dim)  # com a função poster!
            elif dim < 80 and random(10) < 9.5:  # também
                # +3 não permite estrela e casas preenchidas
                poster(x, y, 3, dim * 2, sel_elemento + 3)
            else:  # faz um elemento "sozinho"
                desenha_elemento(x, y, dim, sel_elemento)
    if divisoes == 6:  # uma vez só, na maior grade apenas
        w_olho = dim_total / 18  # dimensão do quadrado olho
        x_olho = random(w_olho, dim_total / 2)
        y_olho = random(dim_total / 2, dim_total - w_olho)
        fill(0)
        square(x_olho, y_olho, w_olho)
        olho(x_olho, y_olho, w_olho * .9)  # definido no verso
```

```
def desenha_elemento(x, y, w, seletor):
    """
    Posicione, ajuste atributos e desenhe os
    elementos de desenho, invocando as funções
    como definidas no verso do pôster:
    estrela() e casinha(), ou, não desenhe nada.
    """
    stroke(0)
    strokeWeight(.5)
    if seletor == 0:  # estrela em preto
        fill(0)
        num_pontas = choice((5, 7, 9))
        ra = w * .35
        rb = choice((w * .25, w * .15, w * .075))
        estrela(x, y, num_pontas, ra, rb)
    elif seletor == 1:  # casinha branca
        fill(255)
        casinha(x, y, w)
    elif seletor == 2:  # casinha preta
        fill(0)
        casinha(x, y, w)
    elif 3 <= seletor < 5:  # 3 ou 4
        noFill()  # casinha sem preenchimento
casinha(x, y, w)
    else: # seletor 5 ou maior
        pass  # não desenha nada!
```