PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO



**RELATÓRIO SOBRE CORES**

LUCAS MACEDO DA SILVA

GOIÂNIA

2020

# Enunciado

**Investigue quais são as primitivas do Processing utilizadas para configuração de cores. Crie no mínimo 6 objetos e utilize-os para mostrar as possibilidades de aplicação destas primitivas na definição de cores de objetos.**

Elabore um relatório e converta-o em PDF.

Segundo o site do processing existem as seguintes primitivas para a definição de cores de objetos:

# background()

Essa função configura a cor de fundo da janela do procesisng. Recebe como parâmetros o valor da cor, as intensidades de cor nos três canais de cor ou a cor na escala de cinza, todas essas opções podem vir junto ou não com a opacidade da cor.

# clear()

Limpa os pixels dentro do buffer. Só funciona em objetos da classe PGraphics.

# colorMode()

Altera a forma que o processing interpreta as cores. Por padrão o modelo de cores é o RGB com valores entre 0 e 255. Com essa função é possível modificar a forma de interpretação e definir os limites de cor usando 3 limites, um para cada canal de cor, e/ou a opacidade das cores.

# fill()

Essa função define as cores usadas para preencher as formas (os objetos). A cor é configurada a partir do modelo de cores usado RGB ou HSB. Pode ser configurada passando o valor da cor no modelo de cor, a intensidade da cor nos três canais ou a cor na escala e cinza, podendo ou não conter o valor da opacidade das cores.

# noFill()

Desativa o preenchimento da forma geométrica. Não recebe nenhum parâmetro.

# stroke()

Configura a cor usada para desenhar as linhas e bordas ao redor das formas. A cor é configurada a partir do modo de cor configurado (RGB ou HSB). Recebe como parâmetros o valor da cor, a intensidade da cor em cada um dos canais de cor ou um valor de intensidade na escala de cinza, junto ou não com o valor da opacidade.

# noStroke()

Essa função desabilita o desenho do traçado, as linhas e bordas ao redor da forma. Ou seja, desabilita a função stroke().

# Criação dos seis objetos e testes das cores em RGB

O seguinte trecho de código cria os seis objetos e configura as cores deles, utilizando diferentes funções do Processing.

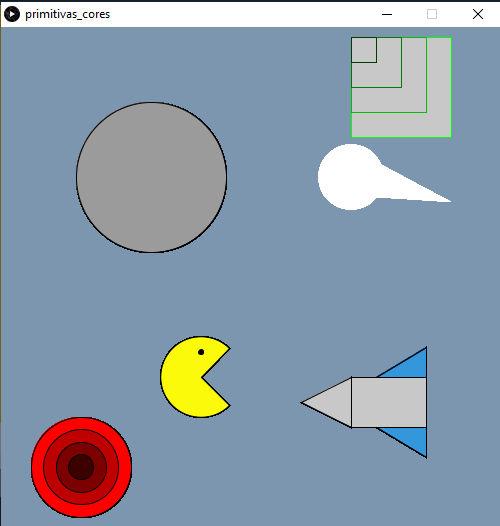
Código 1 - Código para criação dos objetos e configuração das cores no modo de cor RGB

|  |
| --- |
| /\*  \* Aluno: Lucas Macedo da Silva  \* Computação Gráfica  \* Turma: A01  \*  \* Investigação sobre as primitivas de cores do Processing  \* Modo de Cor RGB  \*  \* stroke -> Desenha as linhas da forma (contorno)  \* fill -> Seta a cor da figura  \*/  PGraphics pg;  void setup(){    size(500,500);  background(125,150,175);    pg = createGraphics(250, 250);  // Teste com stroke(rgb)  pg.beginDraw();  pg.stroke(0);    // Teste com fill(rgb) em RGB  pg.fill(155);  pg.circle(150, 150, 150);  pg.endDraw();  image(pg, 0, 0);  }  int value = 0;  void mouseClicked(){  background(125,150,175);    if (value == 0) {  pg.beginDraw();  pg.clear();  pg.endDraw();  value = 1;  } else {  // Teste com stroke(rgb)  pg.beginDraw();  pg.stroke(0);    // Teste com fill(rgb) em RGB  pg.fill(155);  pg.circle(150, 150, 150);  pg.endDraw();  value = 0;  }  image(pg, 0, 0);  }  void draw(){      // Teste com fill(rgb, intensidade)  fill(200, 80);  rect(350, 350, 75, 50);  noFill();    // Teste com fill(r, g, b)  fill(200, 200, 200);  triangle(350, 350, 350, 400, 300, 375);  noFill();    // Teste com fill(r, g, b, intensidade)  fill(50, 150, 220, 80);  triangle(425, 400, 375, 400, 425, 430);  triangle(425, 350, 375, 350, 425, 320);  noFill();    // Teste com fill(r, g, b)  fill(250, 250, 10);  arc(200, 350, 80, 80, PI/4, PI+3\*QUARTER\_PI, PIE);  noFill();    // Teste com fill(rgb)  fill(0);  circle(200, 325, 5);  noFill();    // Teste com fill(rgb) e noStroke()  fill(255);  noStroke();  triangle(375, 170, 375, 135, 450, 175);  circle(350, 150, 65);  noFill();    // Varias itensidades de vermelho  stroke(0);  fill(255,0 ,0);  circle(80, 440, 100);  noFill();  fill(190,0 ,0);  circle(80, 440, 75);  noFill();  fill(125,0 ,0);  circle(80, 440, 50);  noFill();  fill(60,0 ,0);  circle(80, 440, 25);  noFill();    // Varias intesidades de vermelho e verde  fill(200);  noStroke();  stroke(0, 255,0);  square(350, 10, 100);  stroke(0, 190,0);  square(350, 10, 75);  stroke(0, 125,0);  square(350, 10, 50);  stroke(0, 60,0);  square(350, 10, 25);  } |

Fonte: Autoria Própria

Após a execução do código obtém-se o seguinte resultado.

Figura 1 – Resultado da execução do código de criação dos seis objetos



6

5

4

3

2

1

Fonte: Autoria Própria (2020)

Note que os números foram colocados ao lado dos objetos pelo autor para facilitar a discussão. Os objetos serão denominados pelo número que está a seu lado.

A função background foi empregada para configurar o plano de fundo da janela, a cor escolhida foi uma tonalidade de azul escuro. A função stroke foi usada para configurar a cor das bordas dos objetos, foi empregada em todos os objetos a borda na cor preta, exceto o objeto 2 que contém bordas em cores diferentes. A função fill, foi amplamente explorada e é usada para configurar a cor do objeto. A função noFill é utilizada para que apenas o objeto seja colorido, não influenciando a cor dos outros objetos.

**Objeto 1**

O objeto 1 é um círculo e sua cor foi configurada passando um valor na escala de cinza (155). Obtendo um círculo cinza. Ele foi criado como um objeto PGraphics para ser possível a utilização da função clear(). Dessa forma, sempre que ao pressionar o botão direito o mouse o objeto, ao pressionar o botão direito do mouse novamente o objeto aparece, e assim sucessivamente, apenas ilustrando o uso da função clear.

**Objeto 2**

O objeto 2 é formado por diferentes quadrados que vão diminuindo. A cor do objeto foi configurada na escala de cinza (200) com intensidade da cor (80), explorando assim outra forma de configurar cores com o Processing. A borda de cada quadrado foi configurada com diferentes intensidades de verde. Obtendo assim um quadrado com outros quadrados cada vez menores com diferentes cores de verde nas bordas.

**Objeto 3**

O objeto 3, é formado por um círculo e por um triângulo. A cor foi configurada passando um valor na escala de cinza (255) gerando a cor branca, o objeto foi configurado para não apresentar bordas. Gerando um objeto branco sem bordas.

**Objeto 4**

O objeto 4 que aparenta o Pacman foi criado utilizando a função arc, para criar o corpo, e a função circle, para criar o olho. Sua cor foi configurada passando a intensidade de cada cor nos canais vermelho (250), verde (250) e azul (10) obtendo a cor amarela. O olho foi configurado na cor preta passando a intensidade na escala de cinza (0). Ao fim foi obtido um objeto amarelo com um ponto preto dentro, que aparenta o personagem Pacman.

**Objeto 5**

O objeto cinco é um pouco mais complexo, é semelhante à um foguete, ele é formado por três triângulos, ponta e asas laterais, e um retângulo, parte do meio do foguete. A cor foi setada da seguinte forma, a ponta do foguete foi configurada passando o valor na escala de cinza e a intensidade da mesma, a parte do meio do foguete foi configurada passando a intensidade de cada canal de cor (todos em 200) e por fim os triângulos laterais tiveram suas cores configuradas passando a intensidade nos canais de cor vermelho (50), verde (150) e azul (220) bem como a opacidade da cor (80).

**Objeto 6**

O objeto 6 é composto por diversos círculos com diferentes raios. A cor foi configurada passando a intensidade da cor em cada canal. Não foi utilizado os canais de cores azul e verde, apenas o canal de cor vermelho teve sua intensidade variada. Por fim foi obtido um objeto composto por círculos em diferentes intensidades da cor vermelha.

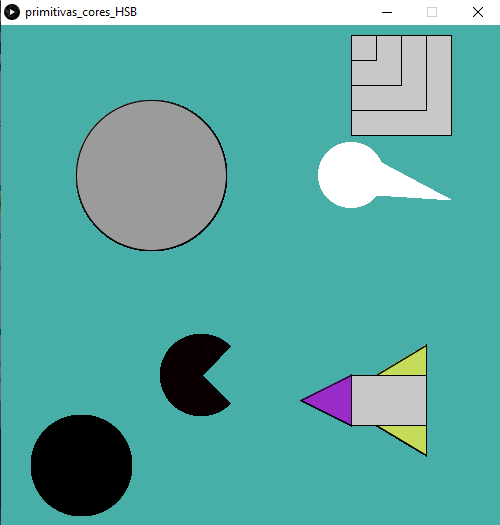
# Teste com o modo de cor HSB

O modo de cor HSB (*Hue-Saturation-Brightnes*, em português: matiz-saturação-brilho) leva em conta a cor-brilho para representar a cor. De forma que o brilho de qualquer tonalidade pura é igual ao brilho do branco puro. As cores nesse sistema são configuradas como uma combinação de matiz, saturação e brilho.

A saturação corresponde a quão vívida a cor será. O brilho pode ser visto como a quantidade de preto na cor. A matiz diz respeito a qual cor monocromática está sendo utilizada.

Foi executado um teste utilizando o código 1 para teste com o espaço de cor HSB. Apenas foi adicionada a linha “colorMode(HSB);” no setup para configurar o espaço de cor HSB. Ao fim foi executado sem modificar nenhum parâmetro do código e foi obtido o seguinte resultado.

Figura 2 – Resultado da execução do código de criação dos seis objetos no espaço de cor HSB



Fonte: Autoria própria (2020)

Percebe-se certa mudança nas cores. Portanto o espaço de cor HSB é um forte candidato a espaço de cor a ser usado nos demais projetos.

# Referências Bibliográficas

PROCESSING. **Background()**. Disponível em: https://processing.org/reference/background\_.html. Acesso em: 02 maio 2020.

PROCESSING. **Clear()**. Disponível em: https://processing.org/reference/clear\_.html. Acesso em: 02 maio 2020.

PROCESSING. **ColorMode()**. Disponível em: https://processing.org/reference/colorMode\_.html. Acesso em: 02 maio 2020.

PROCESSING. **Fill()**. Disponível em: https://processing.org/reference/fill\_.html. Acesso em: 02 maio 2020.

PROCESSING. **NoFill()**. Disponível em: https://processing.org/reference/noFill\_.html. Acesso em: 02 maio 2020.

PROCESSING. **NoStroke()**. Disponível em: https://processing.org/reference/noStroke\_.html. Acesso em: 02 maio 2020.

PROCESSING. **Stroke()**. Disponível em: https://processing.org/reference/stroke\_.html. Acesso em: 02 maio 2020.

SAMPAIO, Luciano de. **O que é espaço de cores?** Disponível em: https://www.tecmundo.com.br/video/2481-o-que-e-espaco-de-cores-.htm. Acesso em: 02 maio 2020.

THOMAS, James. **Teoria da Cor Avançada**: o que é gerenciamento de cores e os designers deveriam saber sobre isso?. Disponível em: https://design.tutsplus.com/pt/articles/advanced-color-theory-what-is-color-management--cms-26307. Acesso em: 02 maio 2020.