```
let food;
let points = 0;
let paused = false;
let menu = true;
let customColor = false;
let snakeColor;

// Mapeamento de cores
const colorMapping = {
  red: [255, 0, 0],
  blue: [0, 0, 255],
  yellow: [255, 255, 0],
  green: [0, 255, 0]
};

function setup() {
  createCanvas(600, 400);
  snakeColor = colorMapping.green; // Cor inicial da cobra (verde)
  snake = new Snake(); // Inicializar a cobra aqui
  food = createVector(random(width), random(height));
}

function draw() {
  background(51);
```

```
if (menu) {
  textSize(50);
  fill(255, 0, 0);
  textAlign(CENTER, CENTER);
  text("SNAKE GAME", width / 2, height / 3);
  textSize(20);
  fill(255);
  text("Press 'P' to play", width / 2, height / 2);
  text("Press 'C' to customize the snake color", width / 2, height / 2 + 50);
} else {
  if (!paused) {
    // Atualizar a posição da cobra para ser afastada pelo cursor
    let target = createVector(mouseX, mouseY);
    let dir = p5.Vector.sub(snake.pos, target).normalize();
    snake.update(dir);

    // Desenhar a cobra
    fill(snakeColor[0], snakeColor[1], snakeColor[2]);
    noStroke();
    snake.show();

    // Desenhar a comida
    if (snakeColor === colorMapping.red) {
```

```
    fill(255, 0, 0); // Vermelho
  } else if (snakeColor === colorMapping.blue) {
    fill(0, 0, 255); // Azul
  } else if (snakeColor === colorMapping.yellow) {
    fill(255, 255, 0); // Amarelo
  } else if (snakeColor === colorMapping.green) {
    fill(0, 255, 0); // Verde
  }
  noStroke();
  ellipse(food.x, food.y, 10, 10);

  // Verificar se a cobra comeu a comida
  if (dist(snake.pos.x, snake.pos.y, food.x, food.y) < 10) {
    // Atualizar a cor da cobra com base no mapeamento de cores
    for (let key in colorMapping) {
      if (colorMapping.hasOwnProperty(key) && snakeColor === colorMapping[key]) {
        let index = Object.keys(colorMapping).indexOf(key);
        let newColor = Object.values(colorMapping)[(index + 1) % Object.keys(colorMapping).length];
        snakeColor = newColor;
        break;
      }
    }
    food = createVector(random(width), random(height));
```

```
      snake.grow();
      points++;
    }

    // Verificar se a cobra bateu na parede
    if (snake.pos.x < 0 || snake.pos.x > width || snake.pos.y < 0 || snake.pos.y > height) {
      snake.reset();
      points = 0;
    }

    // Mostrar a pontuação
    textSize(20);
    fill(255);
    text("Points: " + points, 10, 30);
  } else {
    // Mostrar mensagem de pausa
    textSize(50);
    fill(255, 0, 0);
    textAlign(CENTER, CENTER);
    text("PAUSED", width / 2, height / 2);
  }
 }
}
```

```
function mousePressed() {
  if (!menu) {
    paused = !paused;
  }
}

function keyPressed() {
  if (menu) {
    if (key === "p" || key === "P") {
      snake.reset();
      food = createVector(random(width), random(height));
      menu = false;
    } else if (key === "c" || key === "C") {
      customColor = true;
      menu = false;
    }
  } else {
    if (keyCode === 32) { // Tecla de espaço
      paused = !paused;
    }
  }
}
```

```
function windowResized() {
  resizeCanvas(windowWidth, windowHeight);
}

class Snake {
  constructor() {
    this.pos = createVector(width / 2, height / 2);
    this.vel = createVector(0, 0);
    this.body = [];
  }

  update(dir) {
    // Adicionar um pouco de aleatoriedade aos movimentos da cobra
    dir.rotate(random(-PI / 4, PI / 4));
    this.vel.add(dir);
    this.pos.add(this.vel);
    this.vel.mult(0.5); // Adicionar atrito
    this.body.unshift(this.pos.copy());
    if (this.body.length > 10) {
      this.body.pop();
    }
  }

  show() {
```

```
    fill(snakeColor[0], snakeColor[1], snakeColor[2]);
    noStroke();
    ellipse(this.pos.x, this.pos.y, 20, 20);
    for (let i = 0; i < this.body.length; i++) {
      let b = this.body[i];
      ellipse(b.x, b.y, 20, 20);
    }
  }

  grow() {
    this.body.push(createVector(this.pos.x, this.pos.y));
  }

  reset() {
    this.pos = createVector(width / 2, height / 2);
    this.vel = createVector(0, 0);
    this.body = [];
  }
}
```