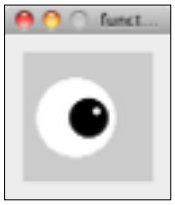
構造化「オリジナル関数の制作」

プログラムの構造化

[function_sample01] 関数の基本1

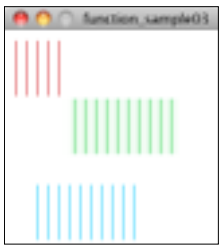


```
void setup(){
  size(100, 100);
  noStroke();
  smooth();
}

void draw(){
  eye();
}

void eye(){
  fill(255);
  ellipse(40, 50, 60, 60);
  fill(0);
  ellipse(50, 50, 30,30);
  fill(255);
  ellipse(56, 45, 6, 6);
}
```

[function_sample02] 関数の基本2。引数の使用

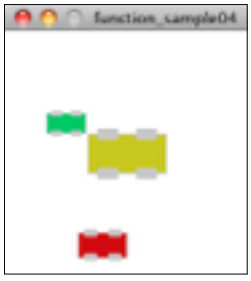
```
void setup(){
  size(100, 100);
  noStroke();
  smooth();
}

void draw(){
  eye(65, 44);
  eye(20, 50);
}

void eye(int x, int y){
  fill(255);
  ellipse(x, y, 60, 60);
  fill(0);
  ellipse(x+10, y, 30,30);
  fill(255);
  ellipse(x+16, y-5, 6, 6);
}
```

[function_sample03] 関数の基本3。複数の引数を使用



```
void setup(){
  size(200, 200);
  smooth();
}

void draw(){
  background(255);
  drawLines(65 ,64 ,10 ,color(20,200,30));
  drawLines(10 ,10 ,5 ,color(200,10,10));
  drawLines(30 ,144 ,10 ,color(20,200,300));
}

void drawLines(int x, int y,int num, color c){
  stroke(c);
  for( int i = 0; i < num; i++){
    line(x+i*10 ,y ,x+i*10, y+50);
  }

}
```

[function_sample04] 関数で複雑な図形を描く



```
void setup() {
  size(200,200);
  smooth();
}

void draw() {
  background(255);
 drawCar(100, 100, 64, color(200,200,0));
 drawCar(50,75,32, color(0,200,100));
 drawCar(80, 175, 40, color(200,0,0));
}

void drawCar(int x, int y, int thesize, color c){

  int offset = thesize/4;
  rectMode(CENTER);
  stroke(200);
  fill(c);
  rect(x,y,thesize,thesize/2);
  fill(200);
  rect(x-offset, y-offset, offset, offset/2);
  rect(x+offset, y-offset, offset, offset/2);
  rect(x-offset, y+offset, offset, offset/2);
  rect(x+offset, y+offset, offset, offset/2);
}
```

オブジェクト指向

classの使用

↓オブジェクト名を定義する

```
Car myCar1, myCar2;

void setup() {
  size(200,200);
  myCar1 = new Car(color(255,0,0),0,100,2);
  myCar2 = new Car(color(0,0,255),0,10,1);
}

void draw() {
  background(255);
  myCar1.move();
  myCar1.display();
  myCar2.move();
  myCar2.display();
}
```

↓それぞれのオブジェクトを初期化

↓それぞれのオブジェクトを操作する

↓クラスの定義

```
class Car {
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car(color tempC, float tempXpos,
                   float tempYpos, float tempXspeed) {
    c = tempC;
    xpos = tempXpos;
    ypos = tempYpos;
    xspeed = tempXspeed;
  }

  void display() {
    stroke(0);
    fill(c);
    rectMode(CENTER);
    rect(xpos,ypos,20,10);
  }

  void move() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = 0;
    }
  }
}
```

←クラス内で使用する変数を定義

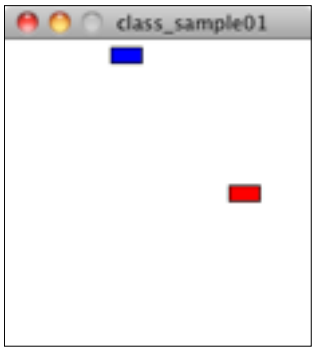
↓コンストラクタ（クラスの初期化）

←クラス内の関数1

←クラス内の関数2

[class_sample01] classの基本1

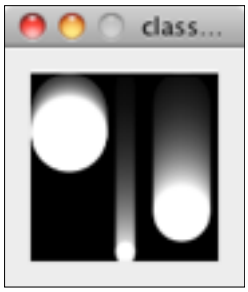
Carクラスを定義し、2つの四角をアニメーションさせる




```
Car myCar1, myCar2; // 2つのオブジェクトを作る

void setup() {
  size(200,200);
   //オブジェクトmyCar1を初期化する
  myCar1 = new Car(color(255,0,0),0,100,2);
   //オブジェクトmyCar2を初期化する
  myCar2 = new Car(color(0,0,255),0,10,1);
}

void draw() {
  background(255);
  myCar1.move(); //オブジェクトmyCar1を操作
  myCar1.display(); //オブジェクトmyCar1を操作
  myCar2.move(); //オブジェクトmyCar2を操作
  myCar2.display(); //オブジェクトmyCar2を操作
}

//クラスの定義ここから
class Car {
   //クラス内で使用する変数を定義する
  color c;
  float xpos;
  float ypos;
  float xspeed;

   //クラス内で使用する変数を初期化する
  Car(color tempC, float tempXpos,
                   float tempYpos, float tempXspeed) {
    c = tempC;
    xpos = tempXpos;
    ypos = tempYpos;
    xspeed = tempXspeed;
  }

   //クラス内の関数その1(オブジェクトの画面表示を担当)
  void display() {
    stroke(0);
    fill(c);
    rectMode(CENTER);
    rect(xpos,ypos,20,10);
  }

   //クラス内の関数その2(オブジェクトの座標移動を担当)
  void move() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = 0;
    }
  }
}
```

[class_sample02] classの基本2

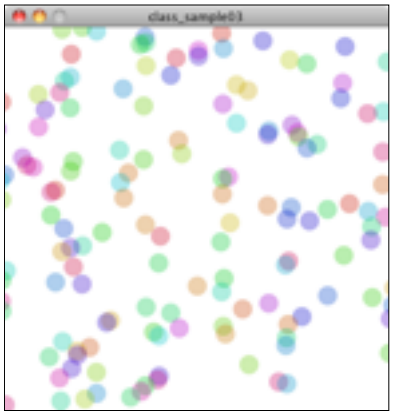
Spotクラスを定義し、3つの円をアニメーションさせる




```
Spot sp1, sp2, sp3; // 3つのオブジェクトの定義

void setup() {
  size(100, 100);
  smooth();
  noStroke();
  sp1 = new Spot(20, 50, 40, 0.5);
  sp2 = new Spot(50, 50, 10, 2.0);
  sp3 = new Spot(80, 50, 30, 1.5);
}

void draw() {
  fill(0, 15);
  rect(0, 0, width, height);
  fill(255);
  sp1.move();
  sp2.move();
  sp3.move();
  sp1.display();
  sp2.display();
  sp3.display();
}

//クラスの定義ここから
class Spot {
  float x, y;
  float diameter;
  float speed;
  // Direction of motion (1 is down, -1 is up)
  int direction = 1;

  Spot(float xpos, float ypos, float dia, float sp) {
    x = xpos;
    y = ypos;
    diameter = dia;
    speed = sp;
  }

  void move() {
    y += (speed * direction);
    if ((y > (height - diameter / 2)) || (y < diameter / 2)) {
      direction *= -1;
    }
  }

  void display() {
    ellipse(x, y, diameter, diameter);
  }
}
```

[class_sample03] classの基本3

クラスと配列の組み合わせで、膨大な数のオブジェクトを同時にアニメーションさせる。



```
int numBall = 200;
Ball[] myBall = new Ball[numBall];

void setup() {
  colorMode(HSB);
  size(400,400);
  smooth();
  frameRate(30);
  for(int i =0; i<myBall.length ; i++){
    myBall[i] = new Ball(color(random(255),255,200),
                         0,i*2,random(20));
  }

}

void draw() {
  background(255);
  for( int i = 0; i<myBall.length; i++){
  myBall[i].move();
  myBall[i].display();
  }
}

//クラスの定義ここから
class Ball {
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Ball(color tempC, float tempXpos,
               float tempYpos, float tempXspeed) {
    c = tempC;
    xpos = tempXpos;
    ypos = tempYpos;
    xspeed = tempXspeed;
  }

  void display() {
    noStroke();
    fill(c,80);
    ellipse(xpos,ypos,20,20);
  }

  void move() {
    xpos = xpos + xspeed;
    if (xpos > width+100) {
      xpos = -100;
    }
  }
}
```

[class_sample04] classの基本4

マウスをクリックした場所に広がる円をそれぞれクラスで定義する。

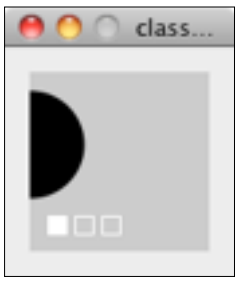
```
Ring[] rings; // Declare the array
int numRings = 50;
int currentRing = 0;

void setup() {
  size(400, 400);
  smooth();
  rings = new Ring[numRings]; // Create the array
  for (int i = 0; i < numRings; i++) {
    rings[i] = new Ring(); // Create each object
  }
}

void draw() {
  background(230);
  for (int i = 0; i < numRings; i++) {
    rings[i].grow();
    rings[i].display();
  }
}

// Click to create a new Ring
void mousePressed() {
  rings[currentRing].start(mouseX, mouseY);
  currentRing++;
  if (currentRing >= numRings) {
    currentRing = 0;
  }
}

//クラスの定義ここから
class Ring {
  float x, y;
  float diameter;
  boolean on = false;

  void start(float xpos, float ypos) {
    x = xpos;
    y = ypos;
    on = true;
    diameter = 1;
  }

  void grow() {
    if (on == true) {
      diameter += 0.5;
      if (diameter > 400) {
        on = false;
      }
    }
  }

  void display() {
    if (on == true) {
      noFill();
      strokeWeight(4);
      stroke(15, 100);
      ellipse(x, y, diameter, diameter);
    }
  }
}
```

[class_sample05] classの基本5

マウスで描いた線の座標が配列に記録され、その線が時間と経過に伴い移動する。

```
int numLines = 500;
MovingLine[] lines = new MovingLine[numLines];
int currentLine = 0;

void setup() {
  size(400, 400);
  smooth();
  frameRate(20);
  for (int i = 0; i < numLines; i++) {
    lines[i] = new MovingLine();
  }
}

void draw() {
  background(204);
  for (int i = 0; i < currentLine; i++) {
    lines[i].display();
  }
}

void mouseDragged() {
  lines[currentLine].setPosition(mouseX, mouseY,
                                 pmouseX, pmouseY);
  if (currentLine < numLines - 1) {
    currentLine++;
  }
}

//クラスの定義ここから
class MovingLine {
  float x1, y1, x2, y2;

  void setPosition(int x, int y, int px, int py) {
    x1 = x;
    y1 = y;
    x2 = px;
    y2 = py;
  }

  void display() {
    x1 += random(-0.1, 0.1);
    y1 += random(-0.1, 0.1);
    x2 += random(-0.1, 0.1);
    y2 += random(-0.1, 0.1);
    strokeWeight(2);
    line(x1, y1, x2, y2);
  }
}
```

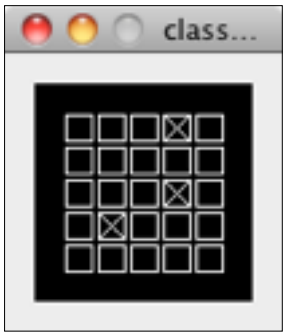
[class_sample06]　classの応用
GUIの提案。左下のボタンをクリックすることで、画面のアニメーションの変化を容易に行うことが出来る。




```
Button button1, button2, button3;
int mode = 1;
//setup関数
void setup() {
  size(100, 100);
  smooth();
  color gray = color(204);
  color white = color(255);
  color black = color(0);
  button1 = new Button(10, 80, 10, gray, white, black);
  button2 = new Button(25, 80, 10, gray, white, black);
  button3 = new Button(40, 80, 10, gray, white, black);
}
//draw関数
void draw() {
  background(204);
  manageButtons();
  noStroke();
  fill(0);
  if (mode == 1) {
    ellipse(0, 40, 60, 60);
  } else if (mode == 2) {
    ellipse(50, 40, 60, 60);
  } else if (mode == 3) {
    ellipse(100, 40, 60, 60);
  }
}
//manageButtons関数(オリジナルの関数)
void manageButtons() {
  button1.update();
  button2.update();
  button3.update();
  button1.display();
  button2.display();
  button3.display();
}
//mousePressed関数
void mousePressed() {
  if (button1.press() == true) {
    mode = 1;
  }
  if (button2.press() == true) {
    mode = 2;
  }
  if (button3.press() == true) {
    mode = 3;
  }
}
//mouseReleased関数
void mouseReleased() {
  button1.release();
  button2.release();
  button3.release();
}
```

```
//クラスの定義
class Button {
  // The x- and y-coordinates
  int x, y;
  int size; // Dimension (width and height)
  color baseGray; // Default gray value
  // Value when mouse is over the button
  color overGray;
  // Value when mouse is over and pressed
  color pressGray;
  boolean over = false; // True when the mouse is over
  // True when the mouse is over and pressed
  boolean pressed = false;

 //クラスのコンストラクタ
  Button(int xp, int yp, int s, color b, color o, color p) {
    x = xp;
    y = yp;
    size = s;
    baseGray = b;
    overGray = o;
    pressGray = p;
  }

  // Updates the over field every frame
  void update() {
    if ((mouseX >= x) && (mouseX <= x + size) &&
        (mouseY >= y) && (mouseY <= y + size)) {
      over = true;
    } else {
      over = false;
    }
  }

  boolean press() {
    if (over == true) {
      pressed = true;
      return true;
    } else {
      return false;
    }
  }

  void release() {
    // Set to false when the mouse is released
    pressed = false;
  }

  void display() {
    if (pressed == true) {
      fill(pressGray);
    } else if (over == true) {
      fill(overGray);
    } else {
      fill(baseGray);
    }
    stroke(255);
    rect(x, y, size, size);
  }
}
```

[class_sample07]　classの応用
チェックボックスをクラスで一括して作る。




```
int numChecks = 25;
Check[] checks = new Check[numChecks];


void setup() {
  size(100, 100);
  int x = 14;
  int y = 14;
  for (int i = 0; i < numChecks; i++) {
    checks[i] = new Check(x, y, 12, color(0));
    x += 15;
    if (x > 80) {
      x = 14;
      y += 15;
    }
  }
}

void draw() {
  background(0);
  for (int i = 0; i < numChecks; i++) {
    checks[i].display();
  }
}

void mousePressed() {
  for (int i = 0; i < numChecks; i++) {
    checks[i].press(mouseX, mouseY);
  }
}
```

```
//クラスの定義
class Check {
  int x, y; // The x- and y-coordinates
  int size; // Dimension (width and height)
  color baseGray; // Default gray value
  // True when the check box is selected
  boolean checked = false;
  Check(int xp, int yp, int s, color b) {
    x = xp;
    y = yp;
    size = s;
    baseGray = b;
  }

  // Updates the boolean variable checked
  void press(float mx, float my) {
    if ((mx >= x) && (mx <= x + size)
              && (my >= y) && (my <= y + size)) {
     // Toggle the check box on and off
    checked = !checked;
    }
  }

  // Draws the box and an X inside
  void display() {
    stroke(255);
    fill(baseGray);
    rect(x, y, size, size);
    if (checked == true) {
      line(x, y, x + size, y + size);
      line(x + size, y, x, y + size);
    }
  }
}
```

[class_sample08]　classの応用

複数のクラスの定義


```
EggRing er1, er2;

void setup() {
  size(100, 100);
  smooth();
  er1 = new EggRing(33, 66, 0.1, 33);
  er2 = new EggRing(66, 90, 0.05, 66);
}

void draw() {
  background(0);
  er1.transmit();
  er2.transmit();
}
```

```
//クラスの定義
class Egg {
  float x, y; // X-coordinate, y-coordinate
  float tilt; // Left and right angle offset
  float angle; // Used to define the tilt
  float scalar; // Height of the egg

  // Constructor
  Egg(int xpos, int ypos, float t, float s) {
    x = xpos;
    y = ypos;
    tilt = t;
    scalar = s / 100.0;
  }

  void wobble() {
    tilt = cos(angle) / 8;
    angle += 0.1;
  }

  void display() {
    noStroke();
    fill(255);
    pushMatrix();
    translate(x, y);
    rotate(tilt);
    scale(scalar);
    beginShape();
    vertex(0, -100);
    bezierVertex(25, -100, 40, -65, 40, -40);
    bezierVertex(40, -15, 25, 0, 0, 0);
    bezierVertex(-25, 0, -40, -15, -40, -40);
    bezierVertex(-40, -65, -25, -100, 0, -100);
    endShape();
    popMatrix();
  }
}
```

```
//クラスの定義
class EggRing {
  Egg ovoid;
  Ring circle = new Ring();

  EggRing(int x, int y, float t, float sp) {
    ovoid = new Egg(x, y, t, sp);
    circle.start(x, y - sp / 2);
  }

  void transmit() {
    ovoid.wobble();
    ovoid.display();
    circle.grow();
    circle.display();
    if (circle.on == false) {
      circle.on = true;
    }
  }
}
```

```
//クラスの定義
class Ring {
  float x, y; // X-coordinate, y-coordinate
  float diameter; // Diameter of the ring
  boolean on = false; // Turns the display on and off

  void start(float xpos, float ypos) {
    x = xpos;
    y = ypos;
    on = true;
    diameter = 1;
  }

  void grow() {
    if (on == true) {
      diameter += 0.5;
      if (diameter > 400) {
        on = false;
      }
    }
  }

  void display() {
    if (on == true) {
      noFill();
      strokeWeight(4);
      stroke(15, 153);
      ellipse(x, y, diameter, diameter);
    }
  }
}
```

[class_sample09]　classの応用

クラスによるパーティクルの表現1

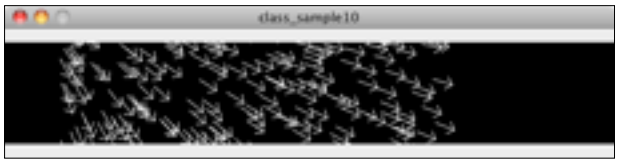

```
int numParticles = 200;
GenParticle[] p = new GenParticle[numParticles];

void setup() {
  size(100, 100);
  noStroke();
  smooth();
  for (int i = 0; i < p.length; i++) {
    float velX = random(-1, 1);
    float velY = -i;
   // Inputs: x, y, x-velocity, y-velocity,
   // radius, origin x, origin y
    p[i] = new GenParticle(width/2, height/2, velX, velY,
                           5.0, width / 2, height / 2);
  }
}

void draw() {
  fill(0, 36);
  rect(0, 0, width, height);
  fill(255, 60);
  for (int i = 0; i < p.length; i++) {
    p[i].update();
    p[i].regenerate();
    p[i].display();
  }
}
```

```
//クラスの定義
class GenParticle extends Particle {
  float originX, originY;

  GenParticle(int xln, int yln, float vxln,
              float vyln, float r, float ox, float oy) {
    super(xln, yln, vxln, vyln, r);
    originX = ox;
    originY = oy;
  }

  void regenerate() {
    if ((x > width + radius) || (x < -radius) ||
        (y > height + radius) || (y < -radius)) {
      x = originX;
      y = originY;
      vx = random(-1.0, 1.0);
      vy = random(-4.0, -2.0);
    }
  }
}
```

```
//クラスの定義
class Particle {
  float x, y; // The x- and y-coordinates
  float vx, vy; // The x- and y-velocities
  float radius; // Particle radius
  float gravity = 0.1;

  Particle(int xpos, int ypos,
                                  float velx, float vely, float r) {
    x = xpos;
    y = ypos;
    vx = velx;
    vy = vely;
    radius = r;
  }

  void update() {
    vy = vy + gravity;
    y += vy;
    x += vx;
  }

  void display() {
    ellipse(x, y, radius*2, radius*2);
  }
}
```

[class_sample10]　classの応用

クラスによるパーティクルの表現2




```
int num = 320;
ArrowParticle[] p = new ArrowParticle[num];
float radius = 1.2;

void setup() {
  size(600, 100);
  smooth();
  for (int i = 0; i < p.length; i++) {
    float velX = random(1, 8);
    float velY = random(-5, -1);
    // Parameters: x, y, x-velocity, y-velocity, radius
    p[i] = new ArrowParticle(0, height / 2,
                                        velX, velY, 1.2);
  }
}

void draw() {
  background(0);
  for (int i = 0; i < p.length; i++) {
    p[i].update();
    p[i].display();
  }
}
```

```
//クラスの定義（パーティクルの形状を担当）
class ArrowParticle extends Particle {
  float angle = 0.0;
  float shaftLength = 20.0;
  ArrowParticle(int ix, int iy, float ivx, float ivy, float ir) {
    super(ix, iy, ivx, ivy, ir);
  }

  void update() {
    super.update();
    angle = atan2(vy, vx);
  }

  void display() {
    stroke(255);
    pushMatrix();
    translate(x, y);
    rotate(angle);
    scale(shaftLength);
    strokeWeight(1.0 / shaftLength);
    line(0, 0, 1, 0);
    line(1, 0, 0.7, -0.3);
    line(1, 0, 0.7, 0.3);
    popMatrix();
  }
}
```

```
//クラスの定義（パーティクルの部分を担当）

class Particle {

  float x, y; // X-coordinate, y-coordinate
  float vx, vy; // X velocity, y velocity
  float radius; // Particle radius
  float gravity = 0.1;

  Particle(int xIn, int yIn, float vxIn, float vyIn, float r) {
    x = xIn;
    y = yIn;
    vx = vxIn;
    vy = vyIn;
    radius = r;
  }

  void update() {
    vy = vy + gravity;
    y += vy;
    x += vx;
  }

  void display() {
    ellipse(x, y, radius*2, radius*2);
  }
}
```