計算機実習 III
第4回：動画の作成2

担当：佐藤

2018年5月1日(火)

前回課題解説

※チャレンジ問題は解説しない．解答例をCoursePowerで確認せよ

2018年5月1日(火)

課題1

```
float r = 250;
int n = 5;
float iAngle = TWO_PI / n;
float angle = iAngle;

void setup() {
  size(600, 600);
  noFill();
  strokeWeight(10);
}

void draw() {
  background(204);
  drawPolygon(width / 2, height / 2);
  angle = (angle + 0.01) % (2 * iAngle);
  if (angle > iAngle) {
    drawLine(width / 2, height / 2, iAngle);
  } else {
    drawLine(width / 2, height / 2, angle);
  }
}
```

静⇔動の切り替えのため周期は内角の2倍

```
void drawPolygon(float cX, float cY) {
  beginShape();
  for (float theta = 0; theta < TWO_PI; theta += iAngle) {
    float x = cX + r * cos(theta);
    float y = cY + r * sin(theta);
    vertex(x, y);
  }
  endShape(CLOSE);
}
```

```
void drawLine(float cX, float cY, float angle) {
  for (int i = 0; i < n; i++) {
    float thetaMIN = i * iAngle;
    float thetaMAX = (i + 1) * iAngle;
    float x = lerp(cX + r * cos(thetaMIN),
                   cX + r * cos(thetaMAX),
                   angle / (iAngle));
    float y = lerp(cY + r * sin(thetaMIN),
                   cY + r * sin(thetaMAX),
                   angle / (iAngle));
    line(cX, cY, x, y);
  }
}
```

辺上の移動位置はlerp()で求められる

課題2

```
float diam = 500;
float theta = 0;
color[] c = { color(255, 0, 0),
              color(0, 255, 0),
              color(0, 0, 255),
              color(255, 255, 0) };
int colorNum = 0;
float[] startAdj = { -HALF_PI,
                     -PI,
                     -PI - HALF_PI,
                     -TWO_PI };
float[] endAdj = { PI + HALF_PI,
                   PI,
                   HALF_PI,
                   0 };
```

4つの要素を持つ配列として色と角度を同じデータ構造で扱い，互いに関連づける

```
void setup() {
  size(600, 600);
  noStroke();
}

void draw() {
  background(204);
  fill(c[(colorNum + 1) % c.length]);
  ellipse(width / 2, height / 2, diam, diam);
  fill(c[colorNum]);
  arc(width / 2, height / 2, diam, diam,
      startAdj[colorNum] + theta,
      endAdj[colorNum] - theta,
      CHORD);
  theta += 0.05;
  if(theta > PI) {
    theta = 0;
    colorNum = (colorNum + 1) % c.length;
  }
}
```

奥の円➔手前の円の順に描画

課題3

```
int n = 12;
float uAngle = TWO_PI / n;
float[] radius = new float[n];
float x = 0, y = 0;
float cX = 0, cY = 0;
int[] sign = new int[n];
float rMax = 250;
float rMin = 100;

void setup() {
  size(600, 600);
  cX = width / 2;
  cY = height / 2;
  noStroke();
  for (int i = 0; i < n; i++) {
    sign[i] = 1;
    radius[i] = random(rMin, rMax + 1);
  }
}
```

- 開始時の移動の向きランダム

```
float r = random(-1, 1);
sign[i] = r >= 0 ? 1: -1;
```

```
void draw() {
  background(255);
  beginShape();
  for (int i = 0; i < n; i++) {
    if (radius[i] < rMin ^ radius[i] > rMax) {
      sign[i] *= -1; // sign[i] = -sign[i];
    }
    radius[i] += sign[i] * 3;
    x = cX + radius[i] * cos(i * uAngle);
    y = cY + radius[i] * sin(i * uAngle);
    curveVertex(x, y);
  }
  // repeat the first 3 vertices
  for (int i = 0; i < 3; i++) {
    x = cX + radius[i] * cos(i * uAngle);
    y = cY + radius[i] * sin(i * uAngle);
    curveVertex(x, y);
  }
  fill(0, 255, 0);
  endShape();
}
```

課題4

```
float r = 0; // radius
float i = 10; // interval

void setup() {
  size(600, 600);
  ellipseMode(RADIUS);
  noStroke();
  mouseX = width / 2;
  mouseY = height / 2;
}

void draw() {
  background(255);
  drawCircleTile();
}
```

```
void drawCircleTile() {
  for (float y = i; y < height; y += 2 * i) {
    for (float x = i; x < width; x += 2 * i) {
      float d = dist(mouseX, mouseY, x, y);
      if (d < 100) {
        r = lerp(17, 1, d / 100);
        fill(#40E0D0);
      } else {
        r = 5;
        fill(#052C03);
      }
      ellipse(x, y, r, r);
    }
  }
}
```

d➔大(r➔小)⇔d➔小(r➔大)

座標系の移動・回転・伸縮・保存と復元

2018年5月1日(火)

translate()

- **translate()**は，座標系を移動する関数
  - 引数：移動後の座標系の原点の座標
  - draw()の中で使用➔**フレームごとにリセット**(次のフレームが始まった時点で原点は(0, 0)に戻る)

例

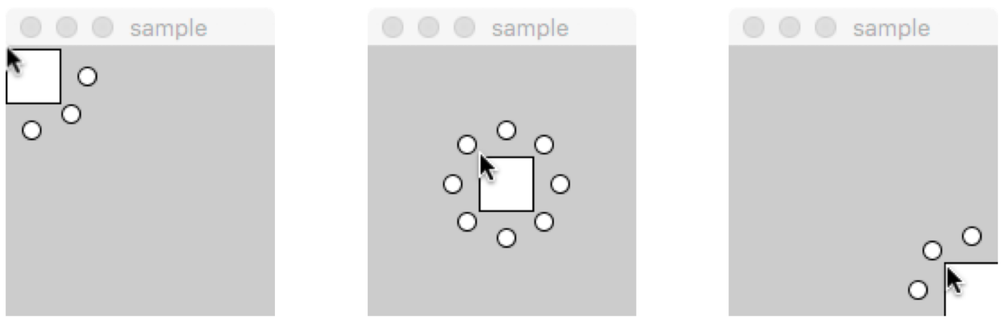
```
float diam = 30;

void setup() {
  size(150, 150);
}

void draw() {
  background(204);
  float x = min(width - diam, mouseX);
  float y = min(height - diam, mouseY);
  translate(x, y);
  rect(0, 0, diam, diam);
  translate(diam / 2, diam / 2);
  drawSatellite(diam);
}
```

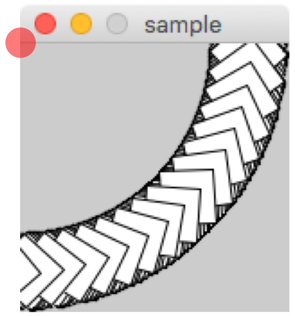
```
void drawSatellite(float radius) {
  for (int i = 0; i < 8; i++) {
    float x = radius * cos(i * QUARTER_PI);
    float y = radius * sin(i * QUARTER_PI);
    ellipse(x, y, diam / 3, diam / 3);
  }
}
```

同じフレームにおいて効果は重なる



rotate()

- `rotate()`は，座標系を回転する関数
  - 引数：回転角[rad]
  - 回転の中心は**原点**

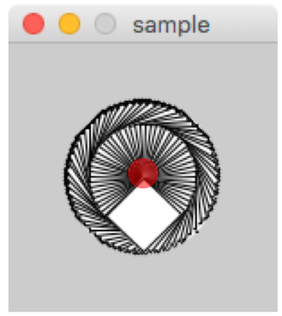


例

```
float angle = 0;

void setup() {
  size(150, 150);
}

void draw() {
  rotate(angle);
  translate(width / 2, height / 2);
  rect(0, 0, 30, 30);
  angle = (angle + 0.1) % TWO_PI;
}
```



例

```
float angle = 0;

void setup() {
  size(150, 150);
}

void draw() {
  translate(width / 2, height / 2);
  rotate(angle);
  rect(0, 0, 30, 30);
  angle = (angle + 0.1) % TWO_PI;
}
```

scale()

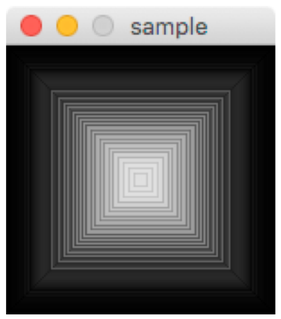
- **scale()**は，座標系を伸縮する関数
  - 引数：伸縮度合い
    - 0(0%)⇔1(100%)(**デフォルト**)⇔2(200%)...
  - draw()の中で使用➔**フレームごとにリセット**(次のフレームが始まった時点でスケールは元の1(100%)に戻る

例

```
float angle = 0;

void setup() {
  size(150, 150);
  rectMode(CENTER);
}

void draw() {
  translate(width / 2, height / 2);
  scale(sin(angle));
  rect(0, 0, 150, 150);
  angle = (angle + 0.01) % TWO_PI;
}
```



pushMatrix(), popMatrix()

- 「pushMatrix();」と「popMatrix();」の間で設定された座標系は，この間に限定される
  - 対象となる関数➔座標系を変更するもの
    - translate(), rotate(), scale()

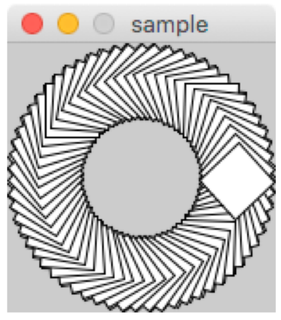
例

```
float angle = 0 ;
float r = 60 - 15 * (sqrt(2) - 1);
float x = 0, y = 0;

void setup() {
  size(150, 150);
  rectMode(CENTER);
  noLoop();
}

void draw() {
  translate(width / 2, height / 2);
  rotate(angle);
  translate(r, 0);
  pushMatrix();
  rotate(QUARTER_PI);
  rect(0, 0, 30, 30);
  popMatrix();
  angle = (angle + 0.1) % TWO_PI;
}
```

```
void mousePressed() {
  redraw();
}
```



動き，方向，スピード

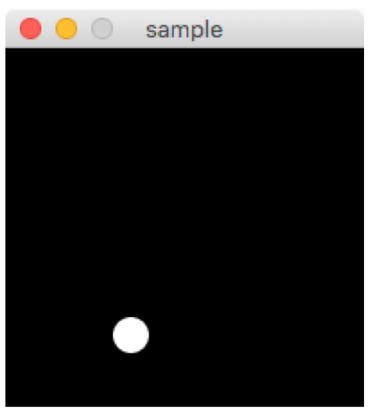
2018年5月1日(火)

移動

例

```
float ballX;
float ballY;
float ballSpeedX = 3;
float ballSpeedY = 3;
float ballDiameter = 20.0;
float ballRadius = ballDiameter / 2;

void setup() {
  size(200, 200);
  ballX = ballRadius;
  ballY = height / 2;
}
```




```
void draw() {
  background(0);

  // draw the ball
  fill(255);
  noStroke();
  ellipse(ballX, ballY, ballDiameter, ballDiameter);

  // update the location of the ball
  ballX += ballSpeedX;
  ballY += ballSpeedY;

  // bouncing at the horizontal edges of the screen
  if (ballX < ballRadius ||
      ballX > width - ballRadius) {
    ballSpeedX = -ballSpeedX;
  }

  // bouncing at the vertical edges of the screen
  if (ballY < ballRadius ||
      ballY > height - ballRadius) {
    ballSpeedY = -ballSpeedY;
  }
}
```

補遺

2018年5月1日(火)

参考

- Processingの公式学習コンテンツ
  - チュートリアル(https://processing.org/tutorials/)
  - サンプルスケッチ(https://processing.org/examples/)
- Processingの書籍(https://processing.org/books/)
  - 初級
    - 『Processingをはじめよう』
  - 中級
    - 『Processing: ビジュアルデザイナーとアーティストのためのプログラミング入門』
  - 上級
    - 『ビジュアライジング・データ ―Processingによる情報視覚化手法』

いずれも作者自身によって書かれた書籍の邦訳版

総合演習Iの準備

2018年5月1日(火)

内容

- 課題
  - 動画スタンプ作成
- 要件
  - 既存のスタンプ(静止画限定．動画は不可)をベースにする
    - 他の受講生とベースにするスタンプが被ってもよい
  - 動画にする
  - ウィンドウの形状は縦長，横長，真四角のいずれでもよい
    - 縦横のサイズはいずれも600以上1400以下にする

評価基準

- 評価項目
  - 参考にしたスタンプの見た目の複雑さに対する完成度
  - 実現したいスタンプの動きの複雑さに対する完成度
  - レポート
- 評価観点
  - 努力度
  - 理解度
  - 独創性

小演習
ベーススタンプの提出

- web検索して作成したいスタンプのベーススタンプ(静止画限定)を決定し，その画像ファイルをダウンロードせよ．その後，画像ファイルを保存・提出せよ
  - 保存場所
    - Z:¥My Documents¥Processing¥computerProgramming3¥xyyyyyyy_05¥
  - 画像ファイル名: xyyyyyyy.zzz
    - x: 自分の青山メールの最初の文字
    - yyyyyyy: 自分の青山メールの2文字目以降の数字の並び
    - zzz: gif, jpg, png, tgaのいずれか
  - 制限提出先
    - 「第4回」➔「総合演習Iベーススタンプ」

100Mバイトまで提出可

- 注
  - 提出可能なベーススタンプは1つのみ
    - 複数のスタンプをベースにすることは不可
    - 作成の過程でベーススタンプ以外のスタンプを参考にすることは可能
  - 提出したベーススタンプの変更は認めない
  - **禁止事項**
    - **私語(周りと相談しない)**
    - **スマホの使用**