メディアアート・プログラミング Ⅰ
動きを生みだす - アニメーションとベクトル

2020年6月12日

東京藝術大学芸術情報センター
田所　淳

アニメーション

- アニメーションをつくるには、すこしづつ変化する画像を、一定間隔で入れ替える必要がある
- パラパラ漫画

- p5.jsでアニメーションを実現するには
- 現在のプログラムをより構造化していく必要がある

アニメーション

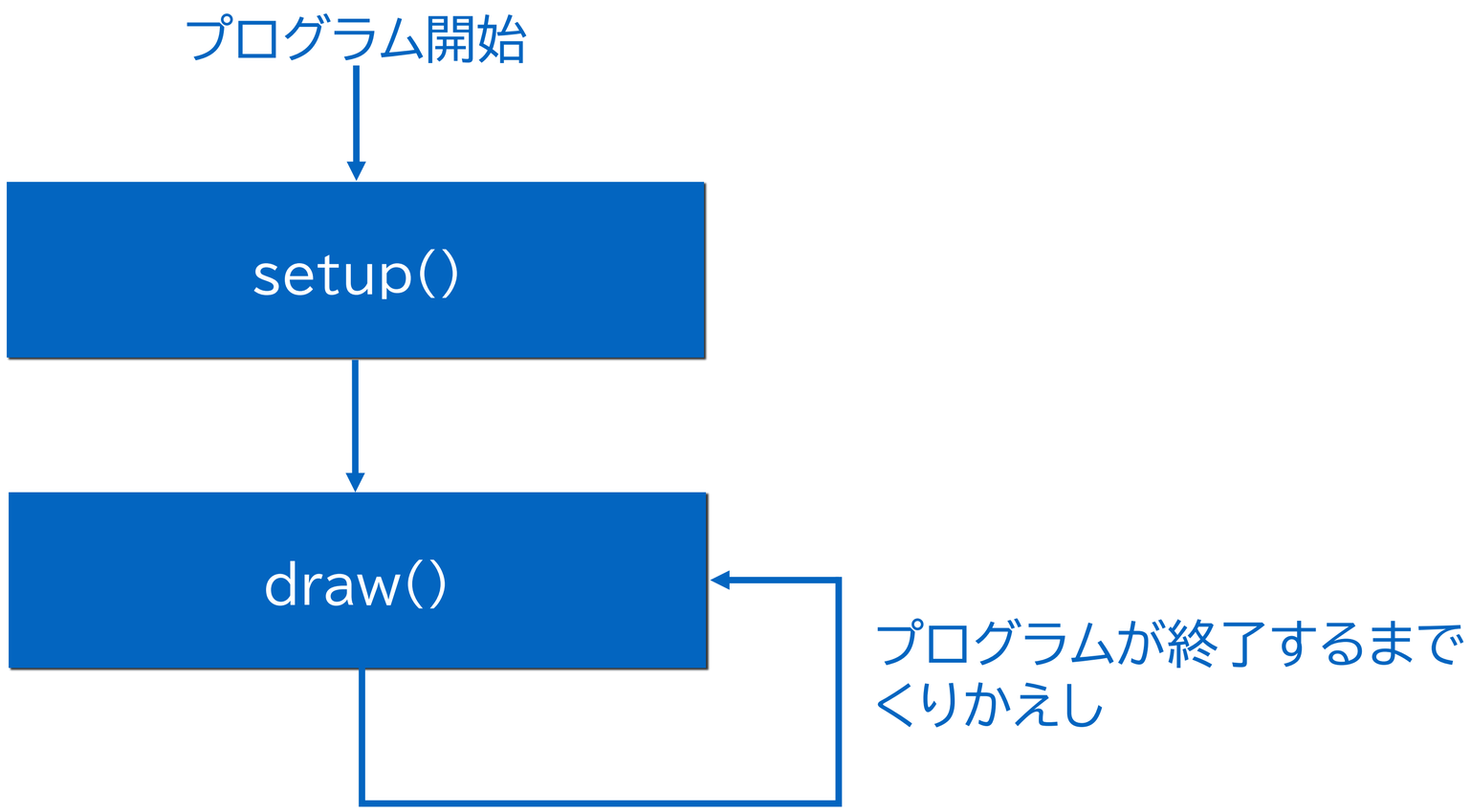
- setup()とupdate()という二つのパートに構造化してアニメーションを実現

- setup() - 初期設定:
- プログラムの起動時に一度だけ実行
- 画面の基本設定やフレームレートなどを設定します。

- draw() - 描画:
- 設定した速さ(フレームレート)でプログラムが終了するまでくりかえし実行されます。

アニメーション

- setup()とdraw() のイメージ



アニメーション

- setup：初期化関数
  - プログラムの最初に、1回だけ実行される処理を記述
  - アニメーションの前準備

- setupの中で行われることの多い処理
  - size：画面のサイズを設定
  - frameRate：画面の書き換え速度を設定
  - colorMode：カラーモードを設定

アニメーション

- draw：メインループ関数
  - プログラムが終了するまでくりかえし
  - ループの中で図形の場所や色、形を操作してアニメーションにする
  - 画面の書き換え頻度はframeRate()関数で設定する

アニメーション

- この構造をp5.jsで記述
- これ以降のプログラムで共通する枠組み

```
// 全体に共通する変数の宣言

function setup() {

    // 初期設定
}

function draw() {

    // 描画
}
```

増殖する円

- この setup() → draw() の構造を実感してみる
- draw() が実行されるたびに、ランダム場所に円（ellipse）を描画してみる

増殖する円

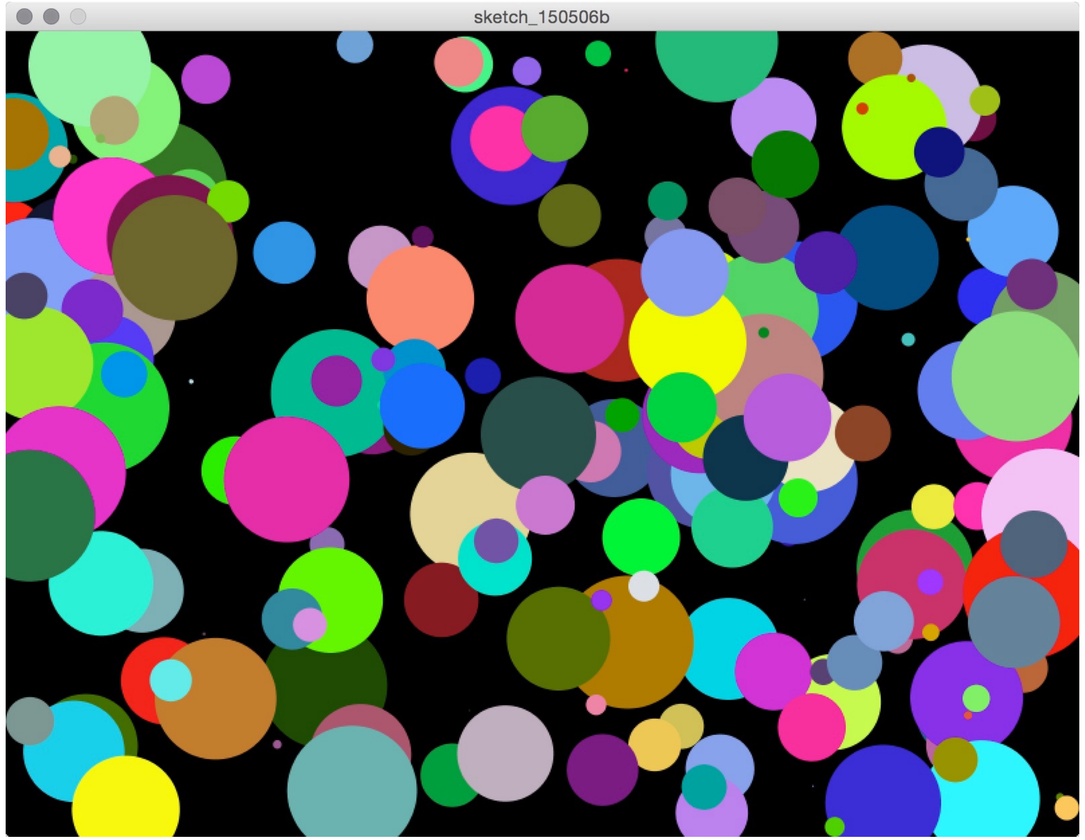
- 構造化されたプログラム例:増殖する円

```
//初期化関数
function setup() {
  createCanvas(windowWidth, windowHeight);
  frameRate(12);   //書き換え頻度の設定
  background(0);
}

//メインループ
function draw() {
  let diameter = random(100);
  noStroke();
  fill(random(255), random(255), random(255));
  ellipse(random(width), random(height), diameter, diameter);
}
```

増殖する円

- 構造化されたプログラム例：増殖する円



アニメーション入門

- アニメーション作成の考えかた
- setup関数で全体に共通の設定を初期化
- draw関数を繰り返す
  - 背景の描画
  - 図形を描く
  - パラメータ(場所、色など)を微妙に変更

アニメーション

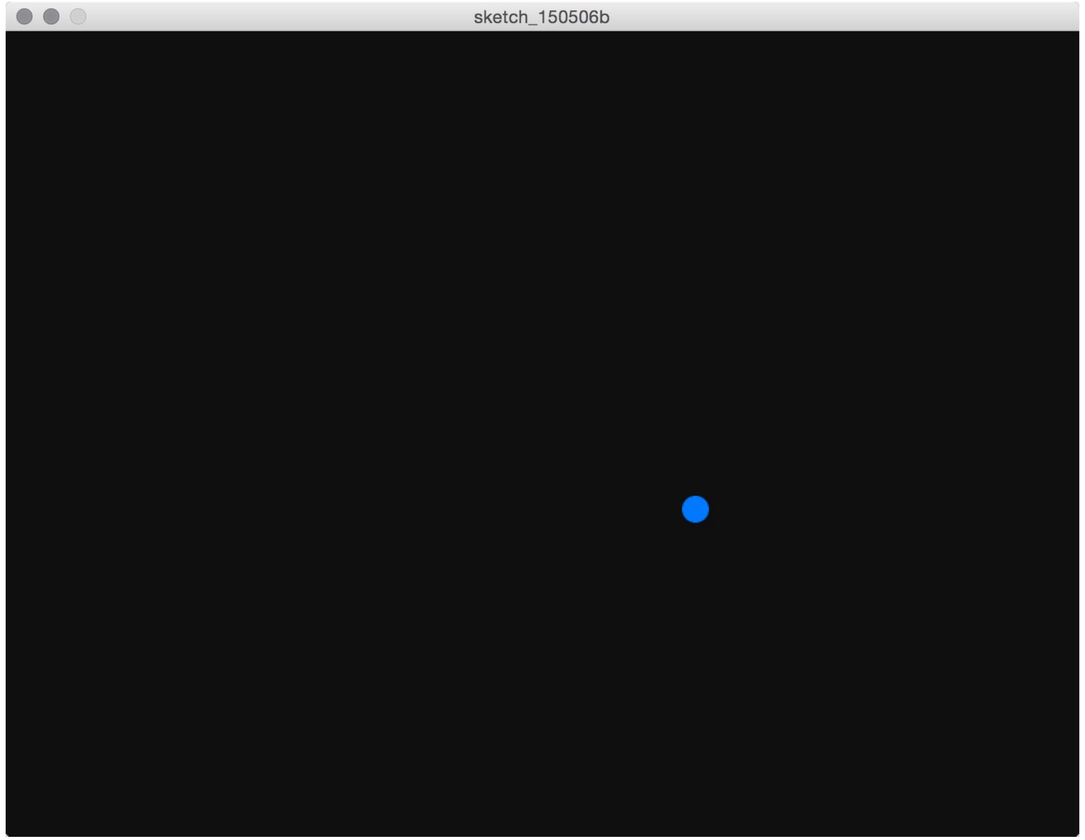
- この仕組みを利用して、円が斜め下に移動するプログラムを作成

```
let locationX, locationY; //円の中心位置を格納する変数
let velocityX, velocityY; //円の速度を格納する変数
function setup() {
  createCanvas(windowWidth, windowHeight); //画面を生成
  frameRate(60); //フレームレート
  locationX = 0; //円の初期位置X
  locationY = 0; //円の初期位置Y
  velocityX = 3; //円の初期位置X
  velocityY = 2; //円の初期位置Y
}

function draw() {
  background(0); //背景を描画
  locationX = locationX + velocityX; //円のX座標を更新
  locationY = locationY + velocityY; //円のY座標を更新
  noStroke(); //枠線なし
  fill(0, 127, 255); //塗りの色
  ellipse(locationX, locationY, 20, 20); //指定した位置に円を描画
}
```

アニメーション

- 円が等速度で移動する



アニメーション入門

- アニメーションの変化するパラメータは座標だけではない
  - 色をアニメーションすることも可能
  - fillの値を変化させれば、色が徐々に変化していく

条件分岐 + アニメーション

条件分岐 + アニメーション

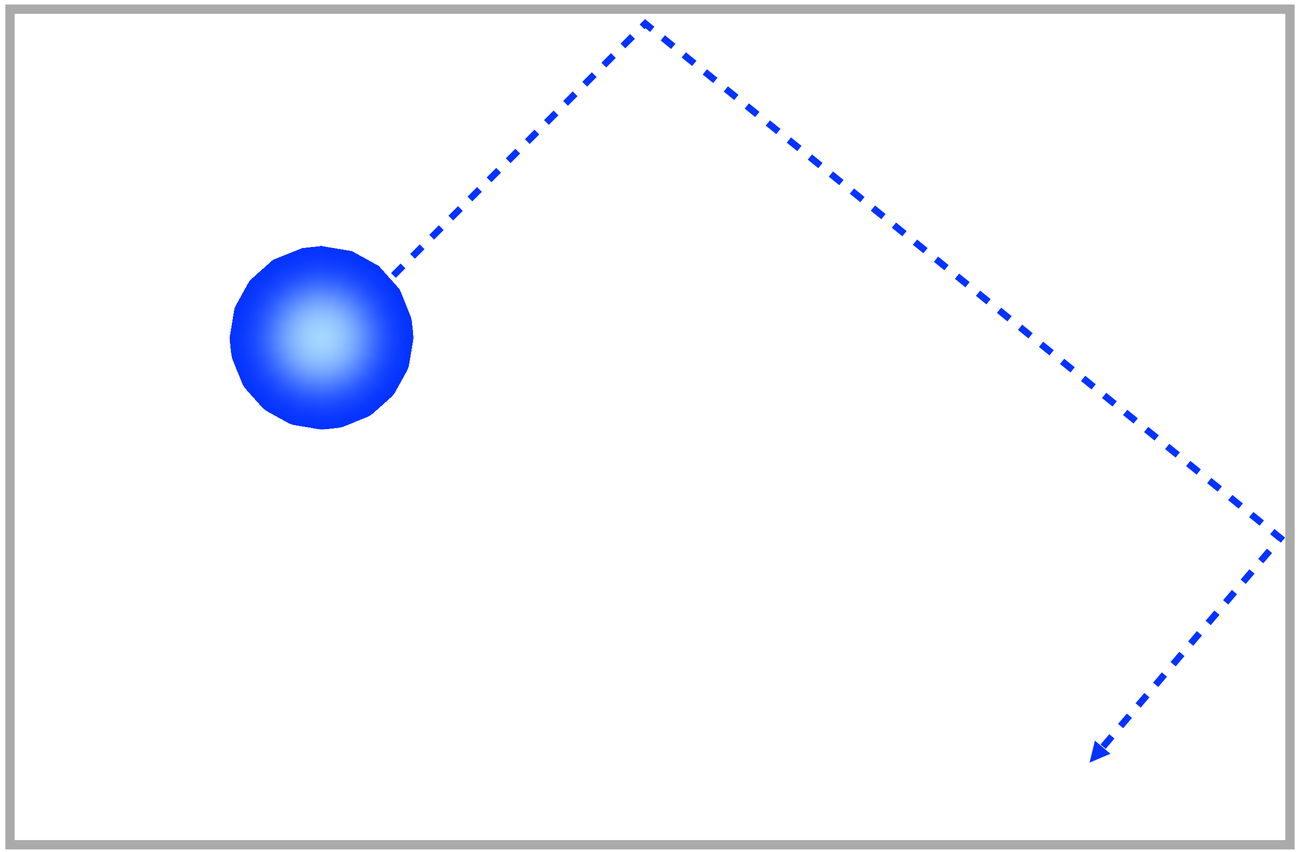
- 画面の端にきたらバウンドする動きを加える

- この動きを実現するには以下の条件を設定する必要あり

- 画面の右端、もしくは画面の左端 → X方向のスピードを反転
- 画面の上端、もしくは画面の下端 → Y方向のスピードを反転

条件分岐 + アニメーション

- バウンドを表現するには？
- x方向のスピードはどうなるのか？
- y方向のスピードはどうなるのか？



条件分岐 + アニメーション

- 条件分岐の構文
- 例 –「もし画面の外にでたら、反対から出現する」
- もし〇〇したら、××せよ：条件分岐命令の典型的な例

```
if([条件式]){
    [条件が正しい場合の処理]
} else {
    [条件が誤っている場合の処理]
}
```

条件分岐

- 条件分岐を適用
  - 条件1：もし左右の端に来たら、X座標方向の速度を反転
  - 条件2：もし上下の端に来たら、Y座標方向の速度を反転

条件分岐

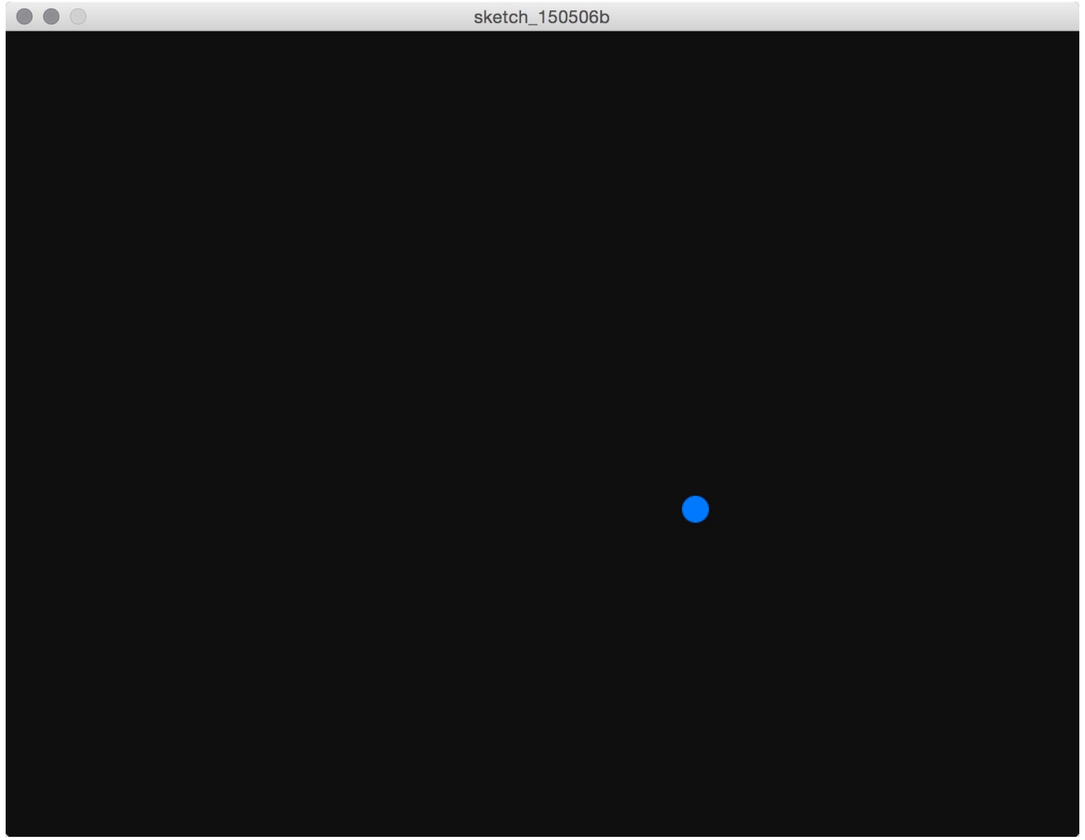
- if文で画面の端でバウンドさせる

```
let locationX, locationY; //円の中心位置を格納する変数
let velocityX, velocityY; //円の速度を格納する変数
function setup() {
  createCanvas(windowWidth, windowHeight); //画面を生成
  frameRate(60); //フレームレート
  locationX = width / 2; //円の初期位置X
  locationY = height / 2; //円の初期位置Y
  velocityX = random(-10, 10); //円の初期速度X
  velocityY = random(-10, 10); //円の初期速度Y
}

function draw() {
  background(0); //背景を描画
  locationX = locationX + velocityX; //円のX座標を更新
  locationY = locationY + velocityY; //円のY座標を更新
  noStroke(); //枠線なし
  fill(0, 127, 255); //塗りの色
  ellipse(locationX, locationY, 20, 20); //指定した位置に円を描画
  //バウンド
  if (locationX < 0 || locationX > width) { //もし画面の左端、または右端に到達したら
    velocityX = velocityX * -1; //X方向のスピードを反転
  }
  if (locationY < 0 || locationY > height) { //もし画面の下端、または上端に到達したら
    velocityY = velocityY * -1; //Y方向のスピードを反転
  }
}
```

条件分岐

- 画面の端でバウンドする動きに!



ベクトル（向きと大きさ）で動きを再定義

ベクトル（向きと大きさ）で動きを再定義

- ここまでの動きの定義の方法
- あまり洗練されたやり方ではなかった

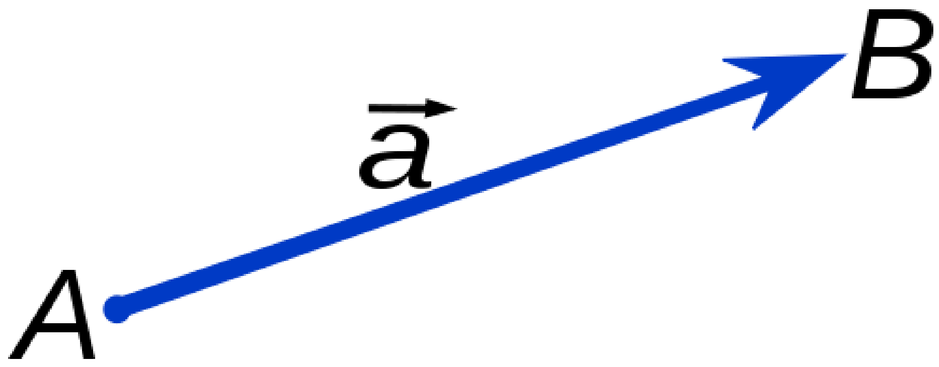
- 位置: locationX と locationY
- 速度: velocityX と velocityY

- それぞれのパラメーターで、常に（x, y）という2つの変数
- もっと整理して定義できないだろうか？

- →「ベクトル」という概念の導入

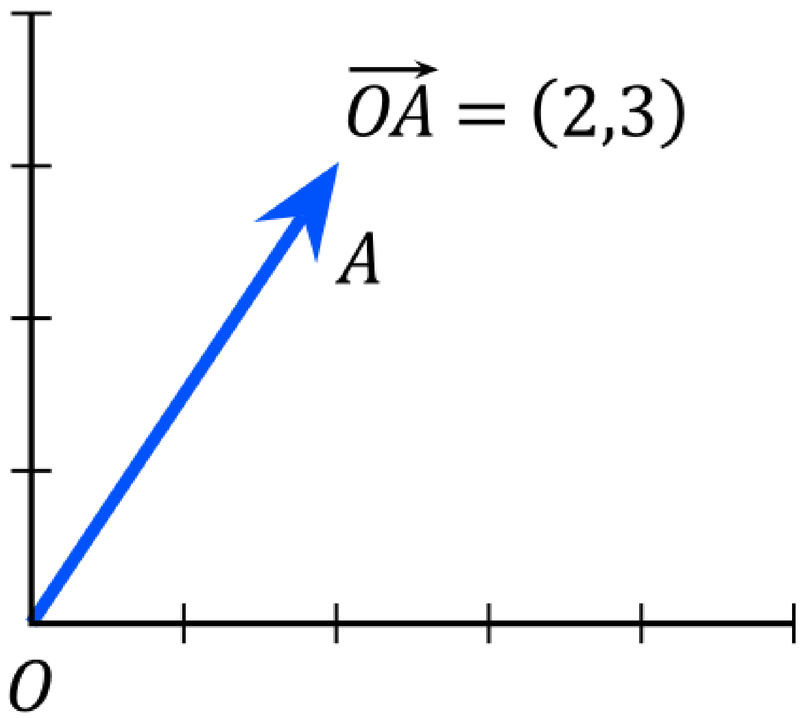
ベクトル（向きと大きさ）で動きを再定義

- ベクトル（vector）
- 幾何学的空間における、大きさと向きを持った量
- 例: 速度、加速度、力など
- 平面上や空間内の矢印としてイメージ



ベクトル(向きと大きさ)で動きを再定義

- 原点(0,0)から(2,3)の座標までのベクトル



ベクトル（向きと大きさ）で動きを再定義

- p5.jsでは、ベクトルを定義するのに適した「p5.Vector」というクラス（属性と動作の型）が用意されている
- 2次元平面でのアニメーション
- p5.Vectorクラスを用いることでとても簡単に整理される

- 画面の端でバウンドするアニメーションを、p5.Vectorを使用して書き直してみる

ベクトル(向きと大きさ)で動きを再定義

- バウンドする動き、ベクトル版

```
let vecLocation; //円の中心の位置ベクトル
let vecVelocity; //円の速度ベクトル
function setup() {
  createCanvas(windowWidth, windowHeight); //画面を生成
  frameRate(60); //フレームレート
  vecLocation = createVector(width / 2, height / 2);
  vecVelocity = createVector(random(-10, 10), random(-10, 10));
}

function draw() {
  background(0); //背景を描画
  vecLocation.add(vecVelocity); //円の座標を更新
  noStroke(); //枠線なし
  fill(0, 127, 255); //塗りの色
  ellipse(vecLocation.x, vecLocation.y, 20, 20); //指定した位置に円を描画
  //バウンド
  if (vecLocation.x < 0 || vecLocation.x > width) { //もし画面の左端、または右端に到達したら
    vecVelocity.x = vecVelocity.x * -1; //X方向のスピードを反転
  }
  if (vecLocation.y < 0 || vecLocation.y > height) { //もし画面の下端、または上端に到達したら
    vecVelocity.y = vecVelocity.y * -1; //Y方向のスピードを反転
  }
}
```

たくさんの図形を同時に動かす

たくさんの図形を同時に動かす

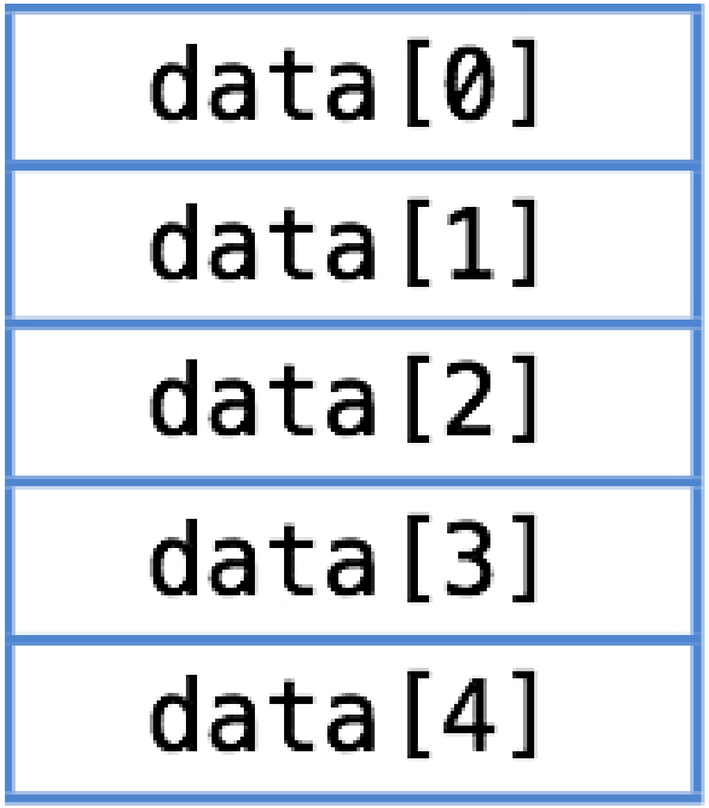
- たくさんの図形を同時に動かすには？

- 図形の数だけ同じコードを書く？
- 動かす図形が、100個、1000個と増えるにつれ破綻する

- こうした際には、配列を使うと良い

たくさんの図形を同時に動かす

- 配列 - データのロッカーのようなイメージ
- 例: float[] data = new float[5];



たくさんの図形を同時に動かす

- 位置(location)、速度(velocity)をそれぞれ配列に変更
- 沢山の位置と速度を扱えるようにしてみる

- 配列と、前回やったくりかえしの構文(for文)とくみあわせる

たくさんの図形を同時に動かす

- 配列の使用

```
let num = 100; //円の数
let vecLocation = []; //円の中心の位置ベクトル
let vecVelocity = []; //円の速度ベクトル

function setup() {
  createCanvas(windowWidth, windowHeight); //画面を生成
  frameRate(60); //フレームレート
  for (let i = 0; i < num; i++) {
    vecLocation[i] = createVector(width / 2, height / 2);
    vecVelocity[i] = createVector(random(-10, 10), random(-10, 10));
  }
}
```

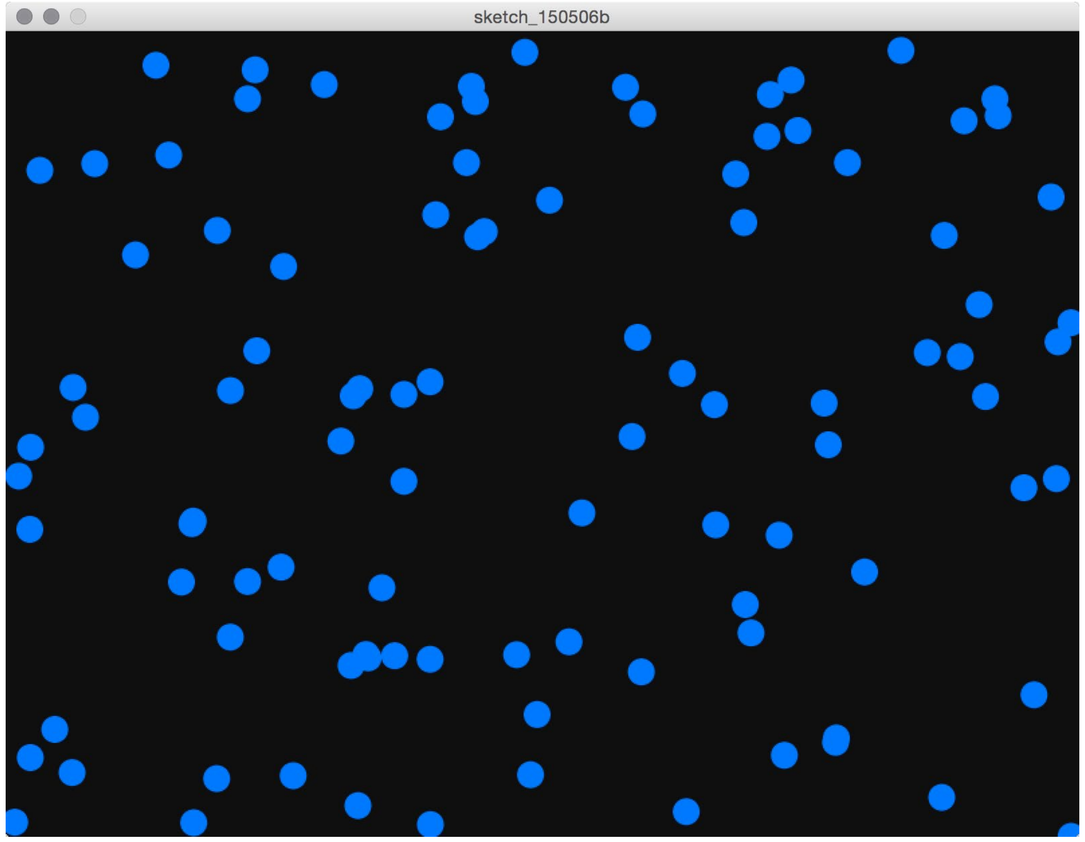
たくさんの図形を同時に動かす

- 配列の使用

```
function draw() {
  background(0); //背景を描画
  noStroke(); //枠線なし
  fill(0, 127, 255); //塗りの色
  //円の数だけくりかえす
  for (let i = 0; i < num; i++) {
    vecLocation[i].add(vecVelocity[i]);
    ellipse(vecLocation[i].x, vecLocation[i].y, 20, 20);
    if (vecLocation[i].x < 0 || vecLocation[i].x > width) {
      vecVelocity[i].x = vecVelocity[i].x * -1;
    }
    if (vecLocation[i].y < 0 || vecLocation[i].y > height) {
      vecVelocity[i].y = vecVelocity[i].y * -1;
    }
  }
}
```

たくさんの図形を同時に動かす

- 100個の円が、同時に動く



たくさんの図形を同時に動かす

- さらにプログラムを工夫して、色と大きさもランダムに

```
let vecLocation = []; //円の中心の位置ベクトル
let vecVelocity = []; //円の速度ベクトル
let diameter = []; //円の直径
let col = []; //円の色
let num = 100;

function setup() {
  createCanvas(windowWidth, windowHeight); //画面を生成
  frameRate(60); //フレームレート
  for (let i = 0; i < num; i++) {
    vecLocation[i] = createVector(width / 2, height / 2);
    vecVelocity[i] = createVector(random(-10, 10), random(-10, 10));
    diameter[i] = random(5, 80);
    col[i] = color(random(255), random(255), random(255), 190);
  }
}
```

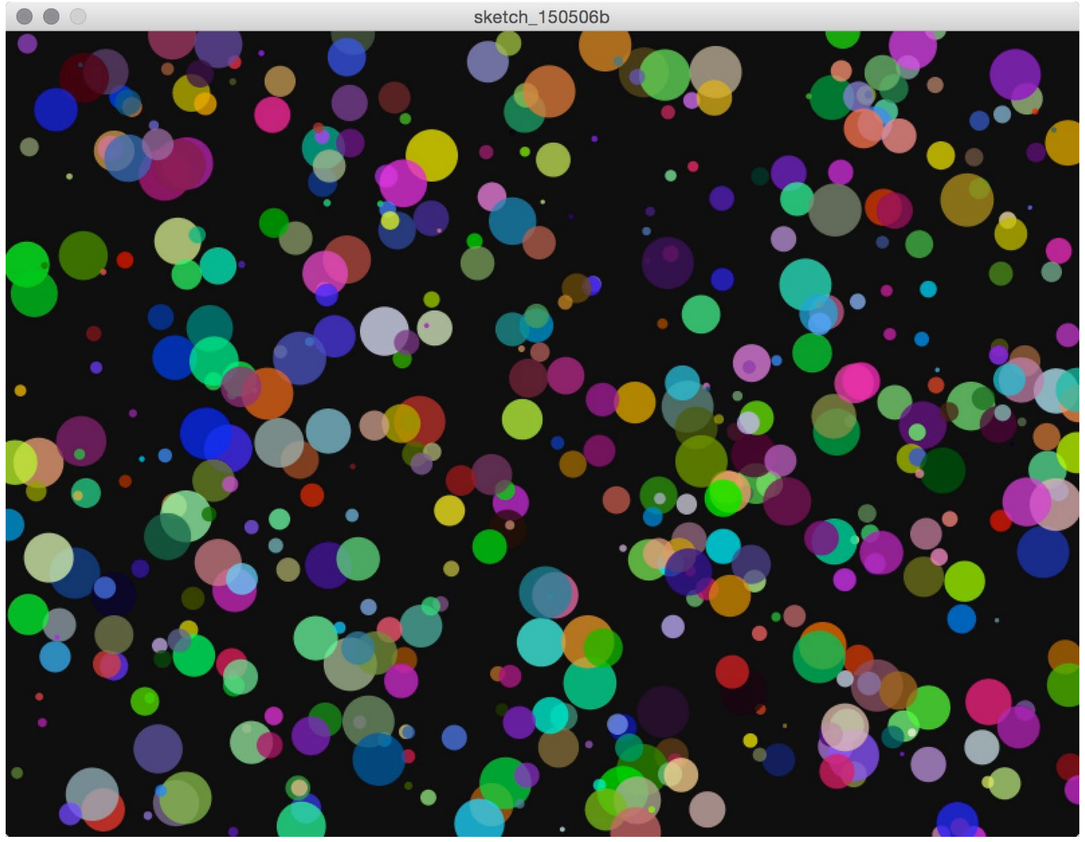
たくさんの図形を同時に動かす

- さらにプログラムを工夫して、色と大きさもランダムに

```
function draw() {
  background(0);
  noStroke();
  for (let i = 0; i < num; i++) {
    fill(col[i]);
    vecLocation[i].add(vecVelocity[i]);
    ellipse(vecLocation[i].x, vecLocation[i].y, diameter[i], diameter[i]);
    if (vecLocation[i].x < 0 || vecLocation[i].x > width) {
      vecVelocity[i].x = vecVelocity[i].x * -1;
    }
    if (vecLocation[i].y < 0 || vecLocation[i].y > height) {
      vecVelocity[i].y = vecVelocity[i].y * -1;
    }
  }
}
```

たくさんの図形を同時に動かす

- 完成!



本日の課題: たくさんの物体の動きを表現する

本日の課題: たくさんの物体の動きを表現する

- 本日解説した内容を踏まえて、たくさんの物体の動きを表現してください
  - アニメーションの原理 setup()とdraw()
  - ベクトル
  - 条件分岐(if)
  - 配列

- 作成した作品は、OpenProcessingに投稿してURLをオンラインフォームから提出