プログラミング基礎

アニメーション
マウス操作への反応
キーボード操作への反応
画像の読み込み

目次

- Processing アニメーションの基本
- マウス操作に反応するプログラム
- キーボード操作に反応するプログラム
- 画像の読み込み
- 画像を使ったパラパラマンガ風アニメーション

Processing アニメーションの基本

- draw 関数は、繰り返し実行される
- 繰り返し少しずつ違うものを描画するだけで、あら簡単！アニメーションになる
- 少しずつ違うように描画するためには、変数を使う

- 例：　次スライド
（講義ページからダウンロード可能です）

例

```
float n = 100; // number of frames per animation cycle

void setup(){
  size(400, 400);
  frameRate(50); // number of frames per second
}

void draw(){
  background(255); // don't forget to clear at each frame

  float r1 = frameCount % n / n;
  rect(0, 0, width * r1, height * r1);

  float r2 = r1 - 0.5;
  ellipse(width * 0.5, height * 0.5, width * r2, height * r2);

  for(int i = 0; i < frameCount % n; i++){
    float y = height / n * i;
    line(0, y, y, y);
  }
}
```

マウス操作に反応するプログラム

- マウスの座標は `mouseX mouseY`
- 少し前のマウスの座標は `pmouseX, pmouseY`
- マウスが押されているかは `mousePressed`
- 「～したとき」に応答する関数を作る
  - `void mousePressed()`
  - `void mouseReleased()`
  - `void mouseMoved()`
  - `void mouseDragged()`
  - `void mouseClicked()`
- 例：次スライド（やはりダウンロード可能）

```
void draw(){
  background(255); // don't forget to clear at each frame

  noFill();
  if(a < width){
    ellipse(cx, cy, a, a);
    a = a + 10;
  }

  fill(0);
  ellipse(bx, by, 30, 30);

  if(mousePressed){
    fill(0);
  }
  else{
    noFill();
  }
  line(mouseX, mouseY, pmouseX, pmouseY);
  ellipse(mouseX, mouseY, 10, 10);
}
```

```
void mouseClicked(){
  a = 10;
  cx = mouseX;
  cy = mouseY;
}
```

```
void mouseDragged(){
  if(grabbed){
    bx = mouseX;
    by = mouseY;
  }
}

void mousePressed(){
  if(dist(mouseX, mouseY, bx, by) < 30){
    grabbed = true;
  }
}

void mouseReleased(){
  grabbed = false;
}
```

キーボードは？

- マウスとほぼ同じ
- 変数（どのキーが押されたか）
  - `key`　　文字キー限定
  - `keyCode` 文字キー以外も利用できる
- 関数
  - `void keyPressed()`
  - `void keyReleased()`
  - `Void keyTyped()`

画像の読み込みと表示
（詳細は次スライド以降を参照）

1. プログラムで使いたい画像ファイルを用意 (gif, jpg, png)
2. プログラムに名前を付けて保存する
3. メニューから Sketch > Add File… から用意した画像を追加する
4. loadImage 関数でプログラムに読み込む
   - Pimage 型の変数に入れる
5. image 関数で描画する

画像の読み込みと表示

（先に前のスライドの1～3をしておくこと）

変数 img の有効範囲に注意

```
PImage img;

void setup(){
  size(400, 400);
  img = loadImage("cat1-a.gif");
}

void draw(){
  background(255);
  image(img, mouseX, mouseY);
  image(img, mouseX, mouseY, img.width / 2, img.height / 2);
}
```

変数の有効範囲

```
void setup(){
  size(400, 400);
  PImage img = loadImage("cat1-a.gif");
}

void draw(){
  background(255);
  image(img, mouseX, mouseY);
  image(img, mouseX, mouseY, img.width / 2, img.height / 2);
}
```

setup の中で変数 img を作ると
draw では使えない

Cannot find anything named “img”

変数の有効範囲（=scope）

- 有効範囲は { … } で区切られる

作るときのルール
1つの有効範囲内では、同じ名前のものを1個作れる

使うときのルール
内から外へ探し、最初に見つかったものを指す

image関数

- image(img, x, y)
- image(img, x, y, width, height)
- img
  - 描画したい PImage
- x, y
  - 描画する場所（左上の点）
- width, height
  - 描画する大きさ
  - 省略した場合は元画像の大きさになる

画像を使った
パラパラマンガ風アニメーション

```
String[] filenames = {"cat1-a.gif", "cat1-b.gif"};
PImage[] images;

void setup(){
  size(95, 111); // ウィンドウのサイズを画像に合わせている
  frameRate(2); // 1秒間にdrawを実行する回数を設定

  images = new PImage[filenames.length];
  for(int i = 0; i < filenames.length; i = i + 1){
    images[i] = loadImage(filenames[i]);
  }
}

void draw(){
  background(255);
  int frame = frameCount % images.length;
  image(images[frame], 0, 0);
}
```

frameCount
= draw 関数が実行された回数

提出方法

- 今までと違うので注意！
- プロジェクトのフォルダを zip 圧縮する
  1. プロジェクトを保存
  2. Tools > Archive Sketch
  3. (名前).zip という名前のファイルが出来る
- zip ファイルをBEEFに提出

課題

- 誰かに捧げるインタラクティブアート
- 要件
  - 自分以外の誰かを想定（複数・集団でも可）
  - インタラクティブである
    - 操作できる
    - 操作していない時も何らかの動きがある
  - アートである
    - アートであると主張する