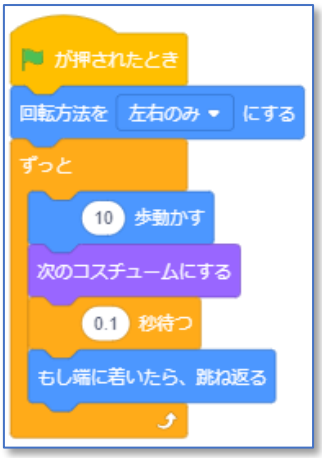
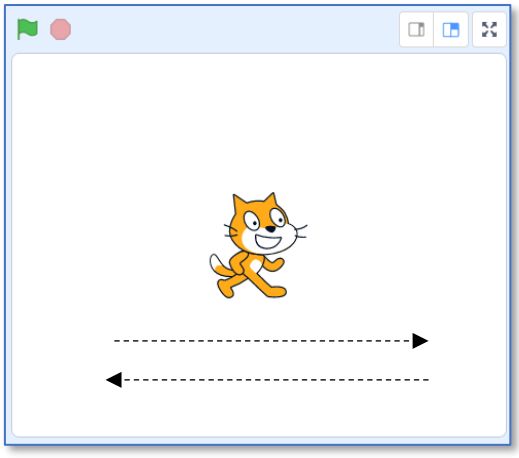
プログラミング練習 (Scratch から Processing へ)

1 ねこ歩きを Processing のコードで表現する

Scratch では Cat のスプライトが左右に歩くプログラムは以下の様に作成します。このプログラムを Processing で表現してみましょう。





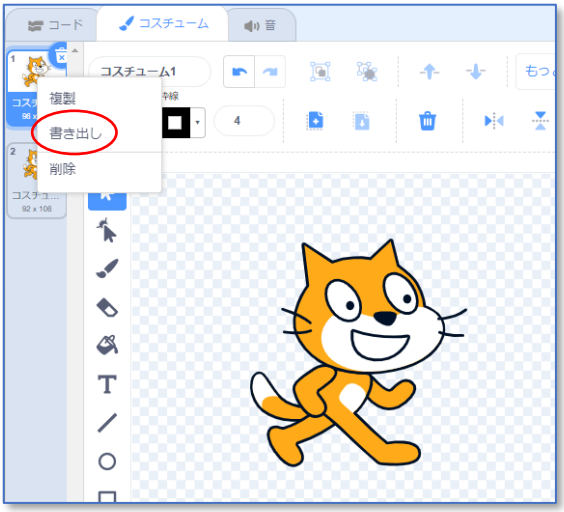
1-1 Scratch(Cat)のイメージデータの取得

Scratch のスプライトのデータをファイルに書き出し、Processing で利用できるようにします。

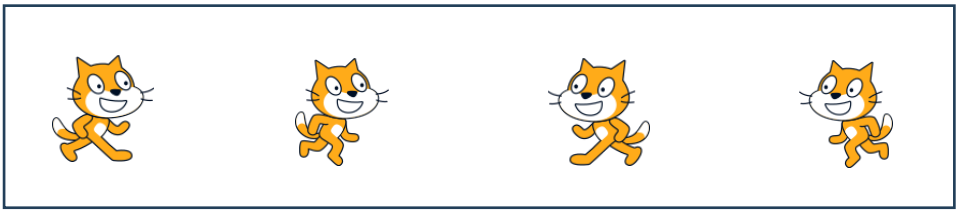
コスチュームを表示し、右クリックで「書き出し」を選び、SVG ファイルとして保存します。

次に、SVG（Scalable Vector Graphics）形式のイメージデータを Processing で利用できる PNG 形式に変換しておきます。PNG 形式のイメージデータは“透明背景”を利用できます。(JPEG 形式に透明背景はない)

変換には無料のサイトや GIMP などの画像処理ソフトを利用します。



＜作成した Cat の PNG ファイル＞

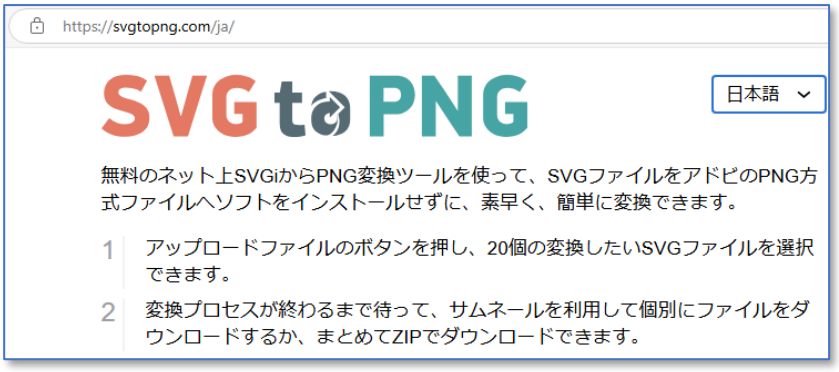


cat1a.png　cat2a.png　cat1b.png　cat2b.png

cat1a と cat2a を左右に反転させた画像

【参考】無料で SVG ファイルを PNG ファイルに変換できるサイトの例

https://svgtopng.com



画像形式　SVG と PNG　（※by Gemini）

SVG と PNG は、どちらも画像形式ですが、その特徴や用途は大きく異なります。

1　**SVG（Scalable Vector Graphics）**

- **ベクター画像:** 線や図形といった数学的な情報で画像を構成するため、拡大縮小しても画質が劣化しません。
- **用途:** ロゴ、アイコン、図形、シンプルなイラストなど、拡大縮小が必要な場合や、高品質で軽量な画像が必要な場合に適しています。

2　**PNG（Portable Network Graphics）**

- **ラスター画像:** ピクセルで構成された画像のため、拡大しすぎると画質が劣化します。
- **用途:** 写真、複雑なイラスト、透明背景が必要な画像など、高画質で細かい表現が必要な場合に適しています。

1-2　ウィンドウと cat の表示

Scratch のステージは、480×360 の大きさで、背景は白となっています。中央には Cat のスプライトが最初から表示されています。

Processing で同じ表示にするプログラムは右のようになります。

```
nekoaruki
//nekoaruki     by Processing4

PImage img1a; // PImage型の変数を宣言
int x,y; //catを表示する座標（左上位置）

void setup() {
  size(480, 360); //Scratchスクリーンサイズ
  img1a = loadImage("cat1a.png"); //96*100サイズ
  x = 240-96/2; //左右を中央に
  y = 180-100/2; //上下を中央に
}

void draw() {
  background(255);  //背景白
  image(img1a, x, y);
}
```

Scratch のステージは中央の座標が(0,0)で、右方向、上方向がそれぞれ正の座標です。また、スプライトはイメージの中央の位置で場所を指定しています。

Processing で作成したウィンドウは左上が(0,0)で、右方向、下方向が正で、表示するイメージは左上の座標で指定しています。

draw()関数は、Scratch の ずっと というブロックと同じはたらきをしています。

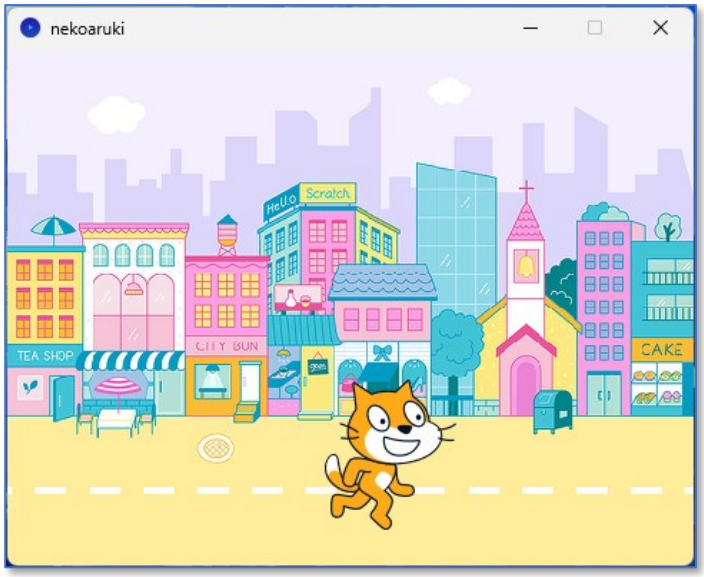
1-3　「ねこ歩き」の作成

背景 Colorful City の中を、ねこが足を動かしながら左右に動く様子を作ります。

Colorful City..png ファイルは Scratch の背景を書き出して作成します。

Cat の **x** 座標を増やすと右に進み、減らすと左に進みます。

進んでいる方向を表す変数として direct を準備し、１で右向き、－１で左向きに進むことにします。



nekoaruki

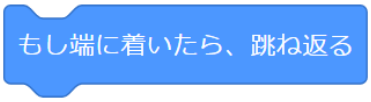
```
//nekoaruki

PImage img1a, img2a, img1b, img2b; // PImage型の変数を宣言
PImage haikei;  //背景用イメージ
int x, y; //catを表示する座標（左上位置）
int disp; //コスチュームの切替用
int direct;  //進む向き 右:1 左:-1
int steps;  //歩数
```

４つの Cat のイメージと背景を PImage 型の変数に読み込んでいます。

```
void setup() {
  size(480, 360); //Scratchスクリーンサイズ
  img1a = loadImage("cat1a.png"); //96*100サイズ
  img2a = loadImage("cat2a.png"); //a-右向き
  img1b = loadImage("cat1b.png"); //b-左向き
  img2b = loadImage("cat2b.png");
  haikei = loadImage("Colorful City.png");
  x = 240-96/2; //左右を中央に
  y = 230;
  disp = 1;  //表示切替用変数
  direct = 1;  //右向き
  steps = 10;  //右に１０歩
}
```

```
void draw() {
  image(haikei,0,0,480,360);  //背景
  if (direct == 1){
    if (disp == 1){image(img1a, x, y);}else{image(img2a, x, y);}   //右向き
  }else{
    if (disp == 1){image(img1b, x, y);}else{image(img2b, x, y);}  //左向き
  }
 x = x + steps;
disp = -disp;  //disp 1 or -1
if (x > 480-95){steps = -10; direct = -1;}  //右端折り返し
if (x < 1){steps = 10; direct = 1;}  //左端折り返し
delay(150);  //0.15秒待ち
}
```

変数 disp の値を切り替えることによって、足の形が異なるイメージを交互に表示しています。

左端、または右端に達したときに、変数 direct の値を切り替え、 

と同じ動きを作っています。

1-4　ねこをジャンプさせる

次のコードを追加させることで、キーボード操作（↑）でねこがジャンプするようになります。

```
void keyPressed() {
  if (keyCode == UP) {
    y = y-150;  //ジャンプ
    image(haikei,0,0,480,360);  //背景
    if (direct == 1){image(img1a, x, y);}else{image(img1b, x, y);}
  y = y + 150;
  }
  if (keyCode == RIGHT) {direct = 1; steps = 10;}  //右向き
  if (keyCode == LEFT) {direct = -1; steps = -10;}  //左向き
}
```

「→」と「←」でねこが進む向きが変わります。

（※ねこがジャンプしない、または方向が変わらない場合、ウィンドウを一度クリックします。）

1-5　風船をキャッチ

Scratch の風船のスプライトのデータを書き出し、色の異なる３つの風船のイメージデータを作成します。ウィンドウの上部に３つの風船を漂わせ、Cat がジャンプしてキャッチする簡単なゲームを作ります。

blue_balloon.png　　yellow_balloon.png　　purple_balloon.png

風船を表示するために、PImage 型の変数を作り、イメージデータを表示します。

Cat と同じ方法で表示する例は以下の通りです。

（作成例１）

```
PImage　balloon1,balloon2,balloon3;
(setup の中で)
balloon1 = loadImage(“blue_balloon.png”);
balloon2 = loadImage(“yellow_balloon.png”);
balloon3 = loadImage(“purple _balloon.png”);
(draw の中で)
image(balloon1,○,○);
　　:
```

（作成例２）PImage 型の配列を使用する方法

```
PImage Balloon[]; //風船用イメージ
String balloons[] =                                    // 文字列配列の宣言
  {"blue_balloon.png", "yellow_balloon.png", "purple_balloon.png"};
```

(setup の中で)

```
Balloon = new PImage[3];  //３個のPImage型配列
for (int i=0; i<3; i++){
  Balloon[i] = loadImage(balloons[i]);
}
```

(draw の中で)

```
for (int i=0; i<3; i++){
  image(Balloon[i], 100*(i+1), 10, 38, 70); //balloon 75×139
}
```

※作成するイメージが多い場合、配列の利用によりプログラムが簡潔になります。

風船の揺らぎを作成するために、 x 座標の変数 balx と得点を記録する変数 point を作成します。

```
int[] balx = {100, 200, 300}; //Balloon[]の x 座標初期値
int point = 0; //得点
```

配列変数 balx に、balx[0] = 100, balx[1] = 200. balx[2] = 300 の値が入ります。

draw 関数の中で、random 関数を使って風船の x 座標を変化させます。

```
for (int i=0; i<3; i++){
  image(Balloon[i], balx[i], 10, 38, 70); //balloon 75×139
  balx[i] += random(-20, 20); //左右の揺らぎ
}
```

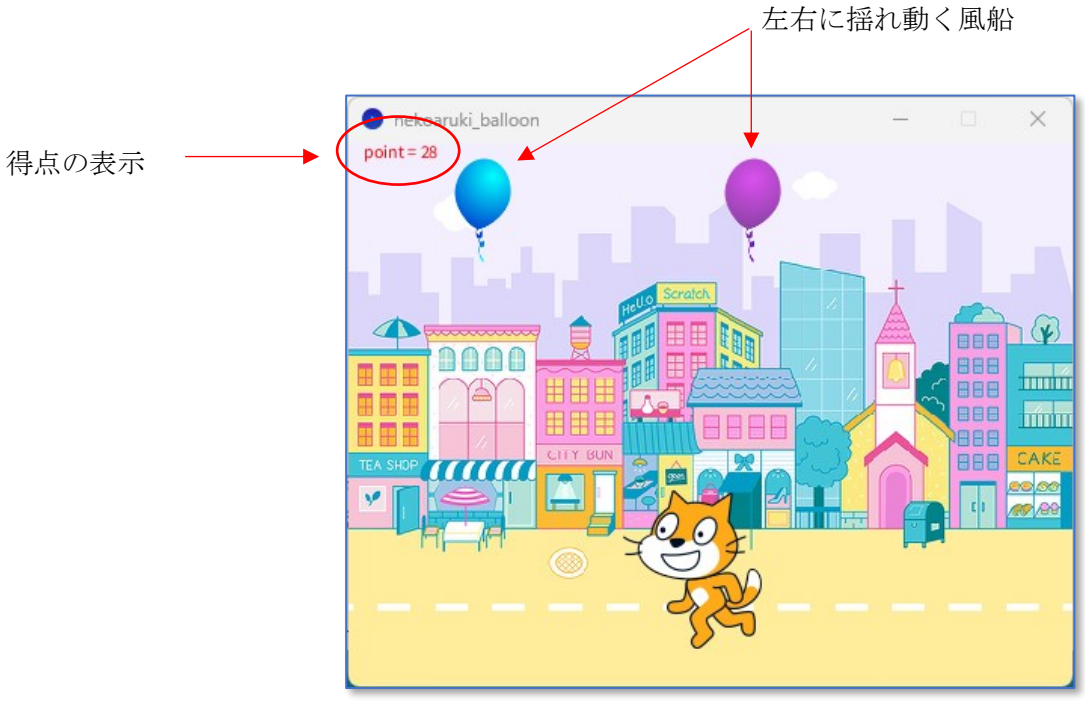
keyPressed()関数の中で、風船と Cat の距離を計算し、100 以下の場合、得点を加算します。

```
dist = sqrt((balx[i] - x)*(balx[i] - x) + (y - 10)*(y - 10));
if(dist <100) {point++;}
```

draw()関数の中で、色を指定して得点を表示させます。

```
fill(255, 0, 0); //赤色
text("point = " + str(point), 10, 10);
```

※text()関数は数値を表示できないので、str()関数で数値 point を文字列に変換しています。



【風船キャッチプログラムリスト（例)】

nekoaruki balloon

```
//nekoaruki_balloon

PImage img1a, img2a, img1b, img2b; // PImage型の変数を宣言
PImage haikei;  //背景用イメージ
PImage Balloon[]; //風船用イメージ
int x, y; //catを表示する座標（左上位置）
int disp; //コスチュームの切替用
int direct;  //進む向き 右:1 左:-1
int steps;  //歩数
float dist;  //Balloonとの距離
String balloons[] =                                  // 文字列配列の宣言
  {"blue_balloon.png", "yellow_balloon.png", "purple_balloon.png"};
int[] balx = {100, 200, 300}; //Balloon[]のｘ座標初期値
int point = 0; //得点

void setup() {
  size(480, 360); //Scratchスクリーンサイズ
  img1a = loadImage("cat1a.png"); //96*100サイズ
  img2a = loadImage("cat2a.png"); //a-右向き
  img1b = loadImage("cat1b.png"); //b-左向き
  img2b = loadImage("cat2b.png");
  haikei = loadImage("Colorful City.png");
  Balloon = new PImage[3];  //３個のPImage型配列
  for (int i=0; i<3; i++){
    Balloon[i] = loadImage(balloons[i]);
  }
  x = 240-96/2; //左右を中央に
  y = 230;
  disp = 1;  //表示切替用変数
  direct = 1;  //右向き
  steps = 10;  //右に１０歩
}

```

```
void draw() {
  image(haikei,0,0,480,360);  //背景
  fill(255,0,0); //赤色
  text("point = " + str(point),10,10);
  for (int i=0; i<3; i++){
    image(Balloon[i],balx[i],10,38,70); //balloon 75×139
    balx[i] += random(-20,20); //左右の揺らぎ
  }
  if (direct == 1){
    if (disp == 1){image(img1a, x, y);}else{image(img2a, x, y);}   //右向き
  }else{
    if (disp == 1){image(img1b, x, y);}else{image(img2b, x, y);}  //左向き
  }
  x = x + steps;
  disp = -disp;  //disp 1 or -1
  if (x > 480-95){steps = -10; direct = -1;}  //右端折り返し
  if (x < 1){steps = 10; direct = 1;}  //左端折り返し
  delay(150);  //0.15秒待ち
}

void keyPressed() {
  if (keyCode == UP) {
    y = y - 150;  //ジャンプ
    image(haikei,0,0,480,360);  //背景
    if (direct == 1){image(img1a, x, y);}else{image(img1b, x, y);}
    for (int i=0; i<3; i++){
      image(Balloon[i],balx[i],10,38,70);
      dist = sqrt((balx[i] - x)*(balx[i] - x) + (y - 10)*(y - 10));
      if(dist <100){point++;}
    }
    y = y + 150;
  }
  if (keyCode == RIGHT) {direct = 1; steps = 10;}  //右向き
  if (keyCode == LEFT) {direct = -1; steps = -10;}  //左向き
}
```

2 オブジェクト指向(Class)を使ったプログラム(参考)

Blloon クラスを作成し、風船オブジェクトが次々と現れるプログラムの例です。

```
//balloon
PImage img1a,img2a,img1b,img2b; // cat 用 PImage 型の変数を宣言
PImage haikei;  //背景用イメージ
PImage balloon1, balloon2, balloon3;  //風船のイメージ
int x,y; //cat を表示する座標（左上位置）
int disp; //コスチュームの切替用
int direct;  //進む向き 右:1 左:-1
int steps;  //歩数
int bal_no;  //風船番号
int point;  //得点

class Balloon {
  int bx, by;
  int bal_no;
  int life = 3;  //life=0 で消える
  Balloon(int bx,int by,int bal_no){
    this.bx = bx;
    this.by = by;
    this.bal_no = bal_no;
  }
  void display(){
    if(bal_no == 1){image(balloon1, bx, by, life*12, life*23);}
    if(bal_no == 2){image(balloon2, bx, by, life*12, life*23);}
    if(bal_no == 3){image(balloon3, bx, by, life*12, life*23);}
    float dist = sqrt((bx-x)*(bx-x)+(by-y)*(by-y));
    if(dist<50){ point++; life--; }  //接近判定
  }
  void move(){
    bx += random(-10,10);  //横の動き
    by += random(-3,2);  //縦の動き
  }
}
ArrayList<Balloon>balloons = new ArrayList<Balloon>();
```

```
void setup() {
  size(480, 360); //Scratch スクリーンサイズ
  img1a = loadImage("cat1a.png"); //96*100 サイズ
  img2a = loadImage("cat2a.png"); //a-右向き
  img1b = loadImage("cat1b.png"); //b-左向き
  img2b = loadImage("cat2b.png");
  haikei = loadImage("Colorful City.png");
  balloon1 = loadImage("balloon1.png"); //75*139 サイズ
  balloon2 = loadImage("balloon2.png");
  balloon3 = loadImage("balloon3.png");
  x = 240-96/2; //左右を中央に
  y = 230;
  disp = 1;  //表示切替用変数
  direct = 1;  //右向き
  steps = 10;  //右に１０歩
  point = 0;  //得点
}

void draw() {
  image(haikei,0,0,480,360);  //背景
  if (random(0,10) < 0.5){
    balloons.add(new Balloon(int(random(10,450)),int(random(50,125)),int(random(1,4))));
  }
  for (int i = balloons.size() - 1; i >=0; i--){
    Balloon bal =balloons.get(i);
    if (bal.by < -15 || bal.life < 1){ balloons.remove(i); }  //枠外風船削除
  }
  for (Balloon bal:balloons){
    bal.display();
    bal.move();
  }
  if (direct == 1){
    if (disp == 1){image(img1a, x, y);}else{image(img2a, x, y);}    //右向き
    }else{
    if (disp == 1){image(img1b, x, y);}else{image(img2b, x, y);}  //左向き
  }
  x = x + steps;
  disp *= -1;  //disp 1 or -1
```

```
    if (x > 480-95){steps = -10; direct = -1;}  //右端折り返し
    if (x < 1){steps = 10; direct = 1;}  //左端折り返し
    delay(150);  //0.15 秒待ち
  }

  void keyPressed() {
    if (keyCode == UP) {
      y = y-180;  //ジャンプ
      image(haikei,0,0,480,360);  //背景
      if (direct == 1){image(img1a, x, y);}else{image(img1b, x, y);}
      for (Balloon bal:balloons){bal.display();}
    y = y + 180;
    }
    if (keyCode == RIGHT) {direct = 1; steps = 10;}  //右向き
    if (keyCode == LEFT) {direct = -1; steps = -10;}  //左向き
  }
```

風船が次々を現れ、ゆらゆらと揺れながら上昇していきます。Cat がジャンプしてタッチすると大きさが小さくなり、３回のタッチでなくなります。

また、上の枠近くまで上昇した風船はなくなります。風船は Balloon クラスのオブジェクトとして生成していて、Scratch のクローンと同じようなはたらきをしています。