情報メディアプロジェクトⅠ：演習課題レポート 1

澤　祐里（21–1–037–0801）

2021 年 06 月 01 日（火）

1　目的

自分が作成したプログラムに用いたアルゴリズムやコードについて文字で説明することで、プログラムをより深く理解する。

2　アルゴリズムの説明

2.1　白塗り円を描くアルゴリズム

円は、360 度あるため、以下の作業を i=0 から 1 ずつ足していきながら 360 回繰り返す。

1. 円を作るために必要な座標を取得するために、i 度の sin と cos を求める。
2. 求めた sin と cos は、あらかじめ決められた半径 R の長さ分だけ掛ける。
3. 図 1 のように sin は y 座標、cos は x 座標にそれぞれ対応するが、double 型で座標の指定に使えないため、int 型に変換する。
4. あらかじめ決められた円の座標を y0 と x0 を sin と cos にそれぞれ足す。
5. sin,cos をそれぞれ配列 enY[i],enX[i] に格納する。
6. 座標 (sin,cos) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。

円の中身を白く塗りつぶすために、上記のループで求めた enX[j] を用いることで、左側の円周から右側の円周に向かって画素値を 255 に変えていく。左側の円周は、円の角度で考えると 91 度から 269 度までが該当し、右側の円周は、0 度から 89 度と 271 度から 359 度までが該当する。このため、円を上半分と下半分で分けて考える。上半分を塗りつぶすには、以下の作業を j=180 から 1 ずつ減らしていきながら、j>90 が true の間は繰り返す。

1. 座標 (enY[j],enX[j]+1 から enX[180-j]-1) が Board.bmp の範囲内の場合はその座標の画素値を 255 にする。
2. 座標 (enY[j],enX[j]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。
3. 座標 (enY[j],enX[180-j]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。

下半分を塗りつぶすには、以下の作業を j=181 から 1 ずつ足していきながら、j<270 が true の間は繰り返す。

1. 座標 (enY[j],enX[j]+1 から enX[540-j]-1) が Board.bmp の範囲内の場合はその座標の画素値を 255 にする。
2. 座標 (enY[j],enX[j]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。
3. 座標 (enY[j],enX[540-j]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。

上記のループで指定できなかった座標 (enY[90],enX[90]) と座標 (enY[270],enX[270]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。

2.2 黒塗り円を描くアルゴリズム

円は、360 度あるため、以下の作業を i=0 から 1 ずつ足していきながら 360 回繰り返す。

1. 円を作るために必要な座標を取得するために、i 度の sin と cos を求める。
2. 求めた sin と cos は、あらかじめ決められた半径 R の長さ分だけ掛ける。
3. 図 1 のように sin は y 座標、cos は x 座標にそれぞれ対応するが、double 型で座標の指定に使えないため、int 型に変換する。
4. あらかじめ決められた円の座標を y0 と x0 を sin と cos にそれぞれ足す。
5. sin,cos をそれぞれ配列 enY[i],enX[i] に格納する。
6. 座標 (sin,cos) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。

円の中身を黒く塗りつぶすために、上記のループで求めた enX[j] を用いることで、左側の円周から右側の円周に向かって画素値を 0 に変えていく。左側の円周は、円の角度で考えると 91 度から 269 度までが該当し、右側の円周は、0 度から 89 度と 271 度から 359 度までが該当する。このため、円を上半分と下半分で分けて考える。上半分を塗りつぶすには、以下の作業を j=180 から 1 ずつ減らしていきながら、j>90 が true の間は繰り返す。

1. 座標 (enY[j],enX[j]+1 から enX[180-j]-1) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。
2. 座標 (enY[j],enX[j]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。
3. 座標 (enY[j],enX[180-j]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。

下半分を塗りつぶすには、以下の作業を j=181 から 1 ずつ足していきながら、j<270 が true の間は繰り返す。

1. 座標 (enY[j],enX[j]+1 から enX[540-j]-1) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。
2. 座標 (enY[j],enX[j]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。
3. 座標 (enY[j],enX[540-j]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。

上記のループで指定できなかった座標 (enY[90],enX[90]) と座標 (enY[270],enX[270]) が Board.bmp の範囲内の場合はその座標の画素値を 0 にする。

2.3 白塗り円と黒塗り円を配置するアルゴリズム

最初に白丸と黒丸の生成数を表す変数、white と black を作成する。while 文を用いて white と black が 10 以上になるまで以下の作業を繰り返し行う。

1. 円の座標 y を表す変数 y0 に (0 から Board.bmp の高さ) までで決められた乱数を入れる。
2. 円の座標 x を表す変数 x0 に (0 から Board.bmp の幅) までで決められた乱数を入れる。
3. 白丸か黒丸のどちらを生成するかを表す変数 borw に (0 か 1) までで決められた乱数を入れる。
4. 円の半径を表す変数 R に (あらかじめ決められた下限値から上限値) までで決められた乱数を入れる。
5. if 文を用いて borw が 0 の時は、白丸を生成するメソッドを実行、borw が 1 の時は黒丸を生成するメソッドを実行。
6. 白丸を生成するメソッドを実行した場合は、その後に、変数 white に 1 を足す。
7. 黒丸を生成するメソッドを実行した場合は、その後に、変数 black に 1 を足す。

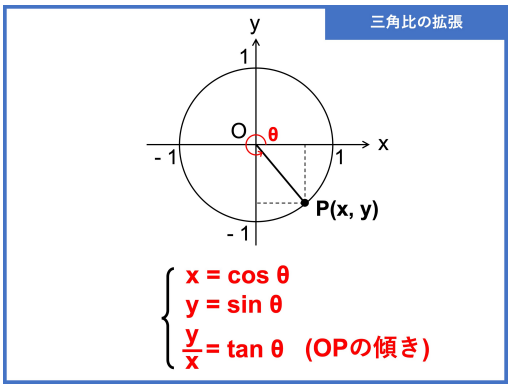

図 1　sin,cos の関係

3　プログラムの説明

3.1　白塗り円を描くメソッドのソースコード

```
  for(int i=0;i<360;i++) {                                    //円を作成するために角度 360 度分、反復する
  b=(R)*Math.sin(Math.toRadians(i));        //0 度から 359 度までをラジアンに変えて、Math.sin メソッドに入れることでその角度の sin を求める
  c=(R)*Math.cos(Math.toRadians(i));        //0 度から 359 度までをラジアンに変えて、Math.Cos メソッドに入れることでその角度の cos を求める
  int Y = (int)b + y0;                //求めた sin と指定した円の座標 y を足して整数化した変数 Y
  int X = (int)c + x0;                //求めた cos と指定した円の座標 x を足して整数化した変数 X
  if(Y>0 && Y<height && X>0 && X<width)    //以下の 1 文を画像の範囲内の時だけ実行
      img.pixel[Y][X] = 0;            //座標（変数 Y, 変数 X）の画素値を 0 にする。（円を作る）
  enX[i] = X;       //変数 X(円周の座標 x) を配列に格納する
  enY[i] = Y;       //変数 Y(円周の座標 y) を配列に格納する
  if(i == 359){    //ループの最後に 1 度だけ実行
      for(int j=180;j>90;j--){     //円の下半分を円周を残して白くする
          for(int p = enX[j]+1;p<enX[180-j];p++){      //左側の円周から右側の円周に向かって、円の中身を白くしていく
              if(enY[j]>0 && enY[j]<height && p>0 && p<width) //Board.bmp の範囲内の時実行
                  img.pixel[enY[j]][p] = 255;
              if(enY[j]>0 && enY[j]<height && enX[j]>0 && enX[j]<width)    //異なる円周が同じ y 座標に存在した時のために、円周を黒くする
                  img.pixel[enY[j]][enX[j]] = 0;
              if(enY[j]>0 && enY[j]<height && enX[180-j]>0 && enX[180-j]<width)
                  img.pixel[enY[j]][enX[180-j]] = 0;
          }
      }
      for(int j=181;j<270;j++){    //円の上半分を円周を残して白くする
          for(int p = enX[j]+1;p<enX[540-j];p++){      //左側の円周から右側の円周に向かって、円の
```

中身を白くしていく

```
                if(enY[j]>0 && enY[j]<height && p>0 && p<width) //Board.bmp の範囲内の時実行
                    img.pixel[enY[j]][p] = 255;
                if(enY[j]>0 && enY[j]<height && enX[j]>0 && enX[j]<width)   //異なる円周が同じ y
```

座標に存在した時のために、円周を黒くする

```
                    img.pixel[enY[j]][enX[j]] = 0;
                if(enY[j]>0 && enY[j]<height && enX[540-j]>0 && enX[540-j]<width)
                    img.pixel[enY[j]][enX[540-j]] = 0;
            }
        }
        //上記のループで指定できなかった円の上と下の頂点を黒くする
        if(enY[90]>0 && enY[90]<height && enX[90]>0 && enX[90]<width)   //Board.bmp の範囲内の時
```

実行

```
            img.pixel[enY[90]][enX[90]] = 0;
        if(enY[270]>0 && enY[270]<height && enX[270]>0 && enX[270]<width)   //Board.bmp の範囲
```

内の時実行

```
            img.pixel[enY[270]][enX[270]] = 0;
    }
}
```

3.2 黒塗り円を描くメソッドのソースコード

```
    for(int i=0;i<360;i++) {     //円を作成するために角度 360 度分、反復する
    b=(R)*Math.sin(Math.toRadians(i));  //0 度から 359 度までをラジアンに変えて、Math.sin メソッドに
```

入れることでその角度の sin を求める

```
    c=(R)*Math.cos(Math.toRadians(i));  //0 度から 359 度までをラジアンに変えて、Math.Cos メソッドに
```

入れることでその角度の cos を求める

```
    int Y = (int)b + y0;     //求めた sin と指定した円の座標 y を足して整数化した変数 Y
    int X = (int)c + x0;     //求めた cos と指定した円の座標 x を足して整数化した変数 X
    if(Y>0 && Y<height && X>0 && X<width)   //以下の 1 文を画像の範囲内の時だけ実行
        img.pixel[Y][X] = 0;     //座標 (変数 Y, 変数 X) の画素値を 0 にする。(円を作る)
    enX[i] = X;              //変数 X(円周の座標 x) を配列に格納する
    enY[i] = Y;              //変数 Y(円周の座標 y) を配列に格納する
    if(i == 359){            //ループの最後に 1 度だけ実行
        for(int j=180;j>90;j--){     //円の下半分を黒くする
            for(int p = enX[j]+1;p<enX[180-j];p++){      //左側の円周から右側の円周に向かって、円の
```

中身を黒くしていく

```
                if(enY[j]>0 && enY[j]<height && p>0 && p<width) //Board.bmp の範囲内の時実行
                    img.pixel[enY[j]][p] = 0;
                if(enY[j]>0 && enY[j]<height && enX[j]>0 && enX[j]<width)
                    img.pixel[enY[j]][enX[j]] = 0;
                if(enY[j]>0 && enY[j]<height && enX[180-j]>0 && enX[180-j]<width)
```

```
                img.pixel[enY[j]][enX[180-j]] = 0;
            }
        }
        for(int j=181;j<270;j++){    //円の上半分を黒くする
            for(int p = enX[j]+1;p<enX[540-j];p++){     //左側の円周から右側の円周に向かって、円の
中身を黒くしていく
                if(enY[j]>0 && enY[j]<height && p>0 && p<width) //Board.bmp の範囲内の時実行
                    img.pixel[enY[j]][p] = 0;
                if(enY[j]>0 && enY[j]<height && enX[j]>0 && enX[j]<width)
                    img.pixel[enY[j]][enX[j]] = 0;
                if(enY[j]>0 && enY[j]<height && enX[540-j]>0 && enX[540-j]<width)
                    img.pixel[enY[j]][enX[540-j]] = 0;
            }
        }
        if(enY[90]>0 && enY[90]<height && enX[90]>0 && enX[90]<width)
            img.pixel[enY[90]][enX[90]] = 0;
        if(enY[270]>0 && enY[270]<height && enX[270]>0 && enX[270]<width)
            img.pixel[enY[270]][enX[270]] = 0;
    }
}
```

3.3 メインプログラムのソースコード

```
    public static void main(String[] args)
{
        String fileName = "Board1.bmp";      //読み込む画像のファイル名を指定
        GImage img= new GImage(fileName);    //Board.bmp の読み込み

        int width = img.getWidth();      //Board.bmp の幅を取得
        int height = img.getHeight();    //Board.bmp の高さを取得

        int Hankeiupper = 50;    //半径の上限値
        int Hankeilower = 5;     //半径の下限値
        int[] enX = new int[360];    //円周の座標 x を入れるための変数
        int[] enY = new int[360];    //円周の座標 y を入れるための変数
        int black = 0;               //黒丸の生成数を入れるための変数
        int white = 0;               //白丸の生成数を入れるための変数

        while(white < 10 || black < 10){     //黒丸と白丸をそれぞれ 10 個以上生成するまでループ
            double b;                        //円を生成するときに使う座標 y を入れるための変数
            double c;                        //円を生成するときに使う座標 x を入れるための変数
            int y0 = (int)(Math.random()*height);    //円の座標 y を指定するための変数 (Board.bmp
```

の高さの範囲でランダムに指定される）

```
            int x0 = (int)(Math.random()*width);     //円の座標 x を指定するための変数（Board.bmp
```

の幅の範囲でランダムに指定される）

```
            int borw = (int)(Math.random()*2);       //黒丸か白丸のどちらを生成するか決める変数（ラ
```

ンダムに決まる）

```
            double R = (Math.random()*Hankeiupper + Hankeilower);    //円の半径を決めるための変
```

数（指定した上限値と下限値の中でランダムに決まる）

```
            if(borw == 0){                                           //白丸を生成する
                for(int i=0;i<360;i++) {                             //円を作成するために角度
```

360 度分、反復する

```
                    b=(R)*Math.sin(Math.toRadians(i));       //0 度から 359 度までをラジアンに変え
```

て、Math.sin メソッドに入れることでその角度の sin を求める

```
                    c=(R)*Math.cos(Math.toRadians(i));       //0 度から 359 度までをラジアンに変え
```

て、Math.Cos メソッドに入れることでその角度の cos を求める

```
                    int Y = (int)b + y0;             //求めた sin と指定した円の座標 y を足して整数
```

化した変数 Y

```
                    int X = (int)c + x0;             //求めた cos と指定した円の座標 x を足して整数
```

化した変数 X

```
                    if(Y>0 && Y<height && X>0 && X<width)    //以下の 1 文を画像の範囲内の時だけ実
```

行

```
                        img.pixel[Y][X] = 0;         //座標（変数 Y, 変数 X）の画素値を 0 にする。（円
```

を作る）

```
                    enX[i] = X;      //変数 X(円周の座標 x) を配列に格納する
                    enY[i] = Y;      //変数 Y(円周の座標 y) を配列に格納する
                    if(i == 359){    //ループの最後に 1 度だけ実行
                        for(int j=180;j>90;j--){     //円の下半分を円周を残して白くする
                            for(int p = enX[j]+1;p<enX[180-j];p++){      //左側の円周から右側の
```

円周に向かって、円の中身を白くしていく

```
                                if(enY[j]>0 && enY[j]<height && p>0 && p<width) //Board.bmp
```

の範囲内の時実行

```
                                    img.pixel[enY[j]][p] = 255;
                                if(enY[j]>0 && enY[j]<height && enX[j]>0 && enX[j]<width)   //
```

異なる円周が同じ y 座標に存在した時のために、円周を黒くする

```
                                    img.pixel[enY[j]][enX[j]] = 0;
                                if(enY[j]>0 && enY[j]<height && enX[180-j]>0 && enX[180-j]<width)
                                    img.pixel[enY[j]][enX[180-j]] = 0;
                            }
                        }
                        for(int j=181;j<270;j++){    //円の上半分を円周を残して白くする
                            for(int p = enX[j]+1;p<enX[540-j];p++){      //左側の円周から右側の
```

円周に向かって、円の中身を白くしていく

```
                                if(enY[j]>0 && enY[j]<height && p>0 && p<width) //Board.bmp
```

の範囲内の時実行

```
                                    img.pixel[enY[j]][p] = 255;
                                if(enY[j]>0 && enY[j]<height && enX[j]>0 && enX[j]<width)   //
```

異なる円周が同じ y 座標に存在した時のために、円周を黒くする

```
                                    img.pixel[enY[j]][enX[j]] = 0;
                                if(enY[j]>0 && enY[j]<height && enX[540-j]>0 && enX[540-j]<width)
                                    img.pixel[enY[j]][enX[540-j]] = 0;
                                }
                            }
                            //上記のループで指定できなかった円の上と下の頂点を黒くする
                            if(enY[90]>0 && enY[90]<height && enX[90]>0 && enX[90]<width)   //Board.bmp
```

の範囲内の時実行

```
                                img.pixel[enY[90]][enX[90]] = 0;
                            if(enY[270]>0 && enY[270]<height && enX[270]>0 && enX[270]<width)   //Board.bmp
```

の範囲内の時実行

```
                                img.pixel[enY[270]][enX[270]] = 0;
                        }
                    }
                    white++;    //白丸を 1 つ生成したことを知らせる
                }
                else{
                    for(int i=0;i<360;i++) {    //円を作成するために角度 360 度分、反復する
                        b=(R)*Math.sin(Math.toRadians(i));  //0 度から 359 度までをラジアンに変えて、
```

Math.sin メソッドに入れることでその角度の sin を求める

```
                        c=(R)*Math.cos(Math.toRadians(i));  //0 度から 359 度までをラジアンに変えて、
```

Math.Cos メソッドに入れることでその角度の cos を求める

```
                        int Y = (int)b + y0;    //求めた sin と指定した円の座標 y を足して整数化した変
```

数 Y

```
                        int X = (int)c + x0;    //求めた cos と指定した円の座標 x を足して整数化した変
```

数 X

```
                        if(Y>0 && Y<height && X>0 && X<width)   //以下の 1 文を画像の範囲内の時だけ実
```

行

```
                            img.pixel[Y][X] = 0;    //座標 (変数 Y, 変数 X) の画素値を 0 にする。(円を作
```

る)

```
                        enX[i] = X;             //変数 X(円周の座標 x) を配列に格納する
                        enY[i] = Y;             //変数 Y(円周の座標 y) を配列に格納する
                        if(i == 359){           //ループの最後に 1 度だけ実行
                            for(int j=180;j>90;j--){    //円の下半分を黒くする
                                for(int p = enX[j]+1;p<enX[180-j];p++){     //左側の円周から右側の
```

円周に向かって、円の中身を黒くしていく

```
                                    if(enY[j]>0 && enY[j]<height && p>0 && p<width) //Board.bmp
```

の範囲内の時実行

```
                                img.pixel[enY[j]][p] = 0;
                            if(enY[j]>0 && enY[j]<height && enX[j]>0 && enX[j]<width)
                                img.pixel[enY[j]][enX[j]] = 0;
                            if(enY[j]>0 && enY[j]<height && enX[180-j]>0 && enX[180-j]<width)
                                img.pixel[enY[j]][enX[180-j]] = 0;
                        }
                    }
                    for(int j=181;j<270;j++){    //円の上半分を黒くする
                        for(int p = enX[j]+1;p<enX[540-j];p++){     //左側の円周から右側の
円周に向かって、円の中身を黒くしていく
                            if(enY[j]>0 && enY[j]<height && p>0 && p<width) //Board.bmp
の範囲内の時実行
                                img.pixel[enY[j]][p] = 0;
                            if(enY[j]>0 && enY[j]<height && enX[j]>0 && enX[j]<width)
                                img.pixel[enY[j]][enX[j]] = 0;
                            if(enY[j]>0 && enY[j]<height && enX[540-j]>0 && enX[540-j]<width)
                                img.pixel[enY[j]][enX[540-j]] = 0;
                        }
                    }
                    if(enY[90]>0 && enY[90]<height && enX[90]>0 && enX[90]<width)
                        img.pixel[enY[90]][enX[90]] = 0;
                    if(enY[270]>0 && enY[270]<height && enX[270]>0 && enX[270]<width)
                        img.pixel[enY[270]][enX[270]] = 0;
                }
            }
            black++;    //黒丸を 1 つ生成したことを知らせる
        }
    }

    String fileName02 = "kadaiF-2";    //出力画像の名称
    String fileType02 = "bmp";         //出力画像の形式
    img.output(fileName02,fileType02); //画像を出力
    fileName02 +="." + fileType02;     //fileName02 に.fileType02 を追加
    System.out.println("Output file:"+fileName02); //文字列"Output file"と fileName02 を出力
}
```

4 演習課題 1

- 図 2 のように、白丸と黒丸がランダムな順番でそれぞれ 10 個以上生成される。
- Board.bmp の範囲を超えた場合は、超えた部分は描写されない。
- 新しく生成された黒丸、または白丸が他の丸に被さる場合は、新しい丸が古い丸の上に重なる。

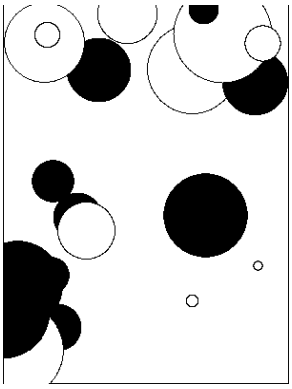

図 2　演習課題 1 の画像

5　考察

- 白丸を生成する部分と黒丸を生成する部分のコードがほとんど同じになったため、引数を変えるだけで白丸と黒丸の生成を同じメソッドとしてまとめることができると考えられる。
- 円の中身を塗りつぶす部分のプログラムで、左側の円周から右側の円周に向かって中身を塗りつぶしていくという方法を選んだが、そもそも円周が同じ y 座標に連続して存在している場合、この方法では、その円周も塗りつぶしてしまうため、結局、円周を描き直す必要がでてきたため、この方法で円の中身を塗りつぶすことはあまり効率が良いとは考えられない。

6　感想

ロボット TA ですぐに答え合わせできるので、プログラムを作ってて楽しいです。