第2回

氏名 林橘平
クラス 総合情報学コース3年
学生証番号 08-192025

□課題2.1 - 2.2節 例1:直線群の描画

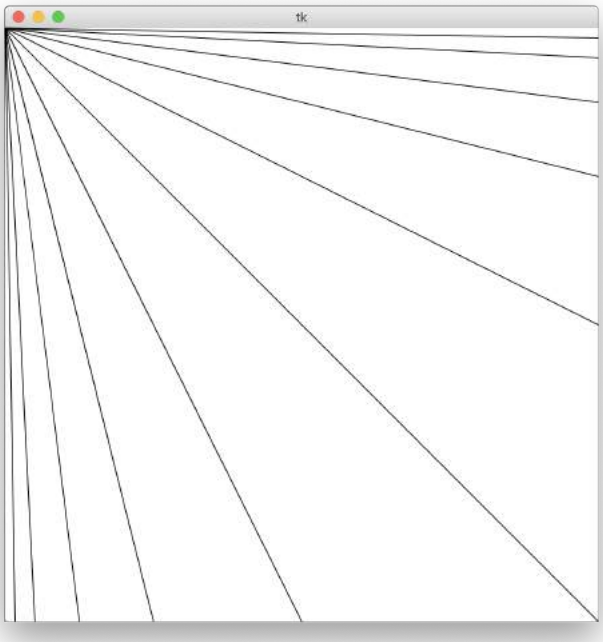
○プログラムリスト

例題のため略

○実行コマンド

```
MBP:Chap02 hayashikippei$ python3 lines.py
```

○実行結果



○考察

create_line()を使えば、始点と終点を指定するだけで滑らかな直線を引いてくれることが確認できた。

□課題2.2 - 2.2節 例2: ピクセルによる直線群の描画

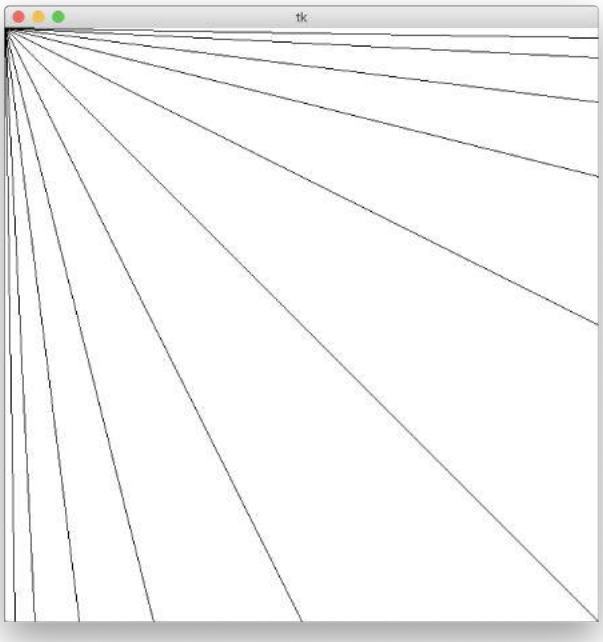
○プログラムリスト

例題のため略

○実行コマンド

```
MBP:Chap02 hayashikippei$ python3 dotLines.py
```

○実行結果



○考察

細い長方形をいくつもつなげることによって直線を描こうとしたプログラム。端から中央にいくに従って長方形の個数が増えてより細かくなるため、 滑らかな直線に見える。中央の線はかなり滑らかでほとんど直線のように見えるが、端の方はかなり荒く、とても滑らかとは言えない。やはり 直線の描写にはcreate_lineメソッドが最適であると考えられる。

□課題2.3 - 2.2節 例3: 円の描画

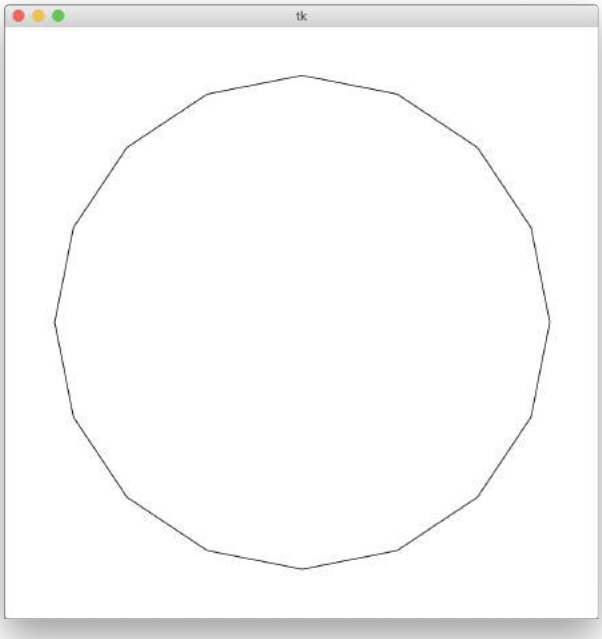
○プログラムリスト

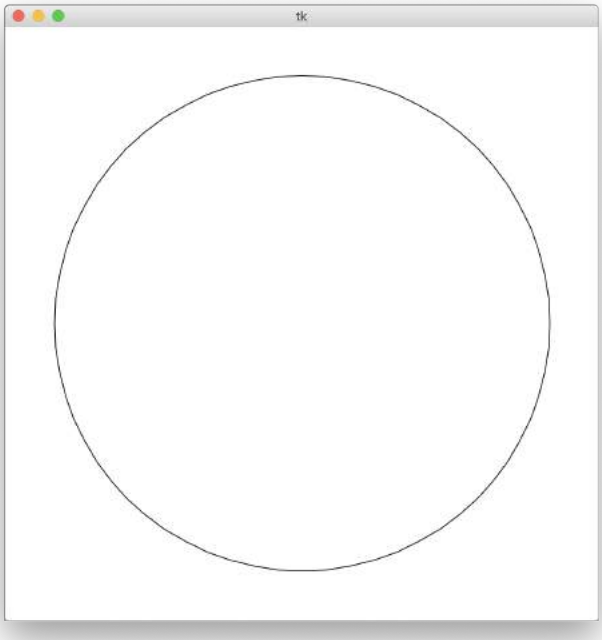
例題のため略

○実行コマンド

```
MBP:Chap02 hayashikippei$ python3 circle.py
# of points -> 16
MBP:Chap02 hayashikippei$ python3 circle.py
# of points -> 64
```

○実行結果





○考察

正多角形の描写により円を描くプログラム。n = 16だとあまりに凸凹が目立ち縁には見えないが、n=64だと凹凸はほとんど目立たず、注意して見ない 限りは円に見える。nを更に大きくすればもちろんより滑らかになるが、その分計算量も増えるため、この程度の数字で十分であると考えられる。

□課題2.4 -2.2節 例4:直線によるカージオイドの描画

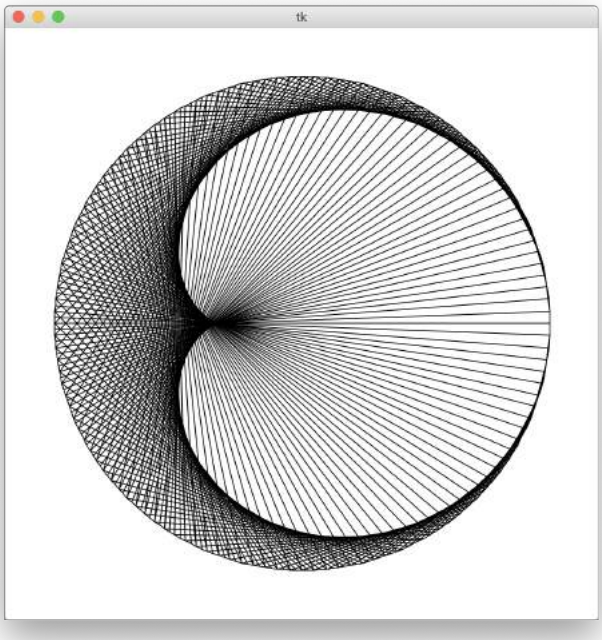
○プログラムリスト

例題のため略

○実行コマンド

```
MBP:Chap02 hayashikippei$ python3 cardioidLine.py
# of points -> 256
```

○実行結果



○考察

プログラム自体は単純であり、circleモジュールで得た正n角形の各頂点PiからP2iに対角線を引いただけである。n = 256とかなり大きくしたため、 カージオイドの形がはっきりと描画されている。

□課題2.5 - 2.2節 例5:円によるカージオイドの描画

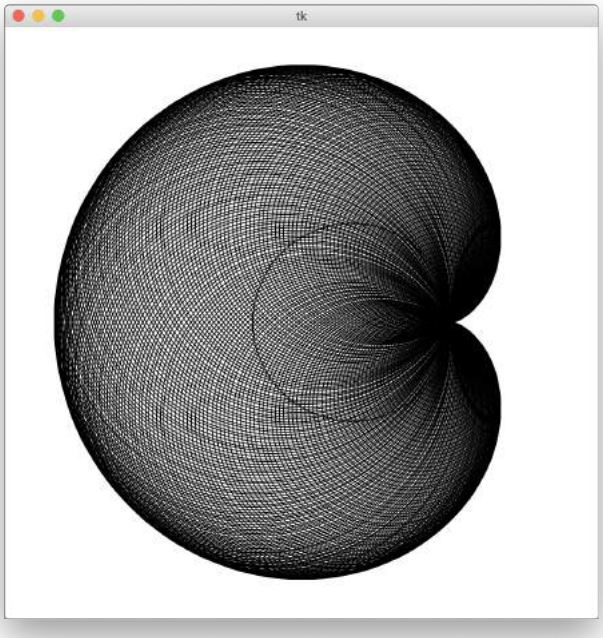
○プログラムリスト

例題のため略

○実行コマンド

```
MBP:Chap02 hayashikippei$ python3 cardioidOval.py
# of points -> 256
```

○実行結果

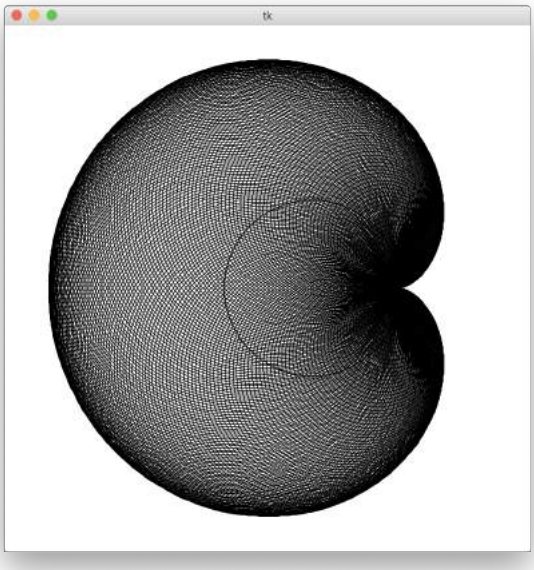


○考察

circleモジュールで得た正n角形の各頂点において基点(0番頂点)からの距離rを求め、create_ovalメソッドによりその頂点を中心とした一辺rの正方形に内接する楕円 （この場合だと正方形なので円)を描く。これは当然基円上に中心を持ち、更に半径がrのため必ず基点を通る。これを各頂点に行ってカージオイドを描く。
基円上に中心を持ち、基点を通る円群は以下のようにしても描ける。circleモジュールを変更してnの値をn = 256に固定した。そして、各頂点の座標と基点からの距離を求めてから それらを中心、半径とする円を描いた。実行結果は下の画像の通りである。
円を描く際にtkinterのモジュールを使わず正256角形を描いているため、 上に比べると所々線が歪んでいる。

```
from tkinter import *
import math

W, H = (600, 600)

def circle(cen= (300,299), r = 250 ,n = 256):
    p = []
    for i in range(n):
        t = 2 * math.pi * i / n
        p.append((r*math.cos(t)+cen[0], r*math.sin(t) + cen[1]))
    return tuple(p)

def circle_display(canvas, points):
    for i in range(len(points)):
        j = (i + 1) % len(points)
        canvas.create_line(points[i], points[j])
```

```
def display(canvas, points):
    for i in range(len(points)):
        r =((points[i][0] - points[0][0]) ** 2 + (points[i][1] - points[0][1])**2)**0.5
        p = circle((points[i][0], points[i][1]),r)
        circle_display(canvas, p)

def main():
    root = Tk()                         # ルートフレームの作成
    canvas = Canvas(root, width = W, height = H, highlightthickness=0)
                                # canvasの作成
    canvas.pack()                       # canvasの配置確定
    points = circle((350, 299),100)
    circle_display(canvas, points)
    display(canvas, points)                     # 描画関数 (display) の呼出
    root.mainloop()                     # ルートフレームの実行ループ開始

if __name__ == '__main__':                  # 起動の確認 (コマンドラインからの起動)
  main()                            # main関数の呼出
```



□課題2.6 - 2.2節 章末課題:ダイアモンドパターンの描画

○プログラムリスト

```
from tkinter import *
import circle

W, H = (600, 600)

def display(canvas, points):
    for i in range(len(points)):
        for j in range(len(points) -3):
            h = (i + j + 2) % len(points)
            canvas.create_line(points[i], points[h])

def main():
    root = Tk()                         # ルートフレームの作成
```

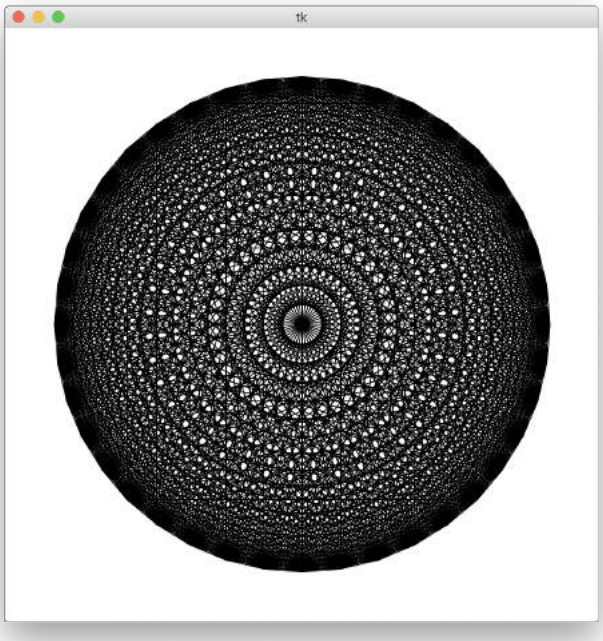
```
    canvas = Canvas(root, width = W, height = H, highlightthickness=0)
                                # canvasの作成
    canvas.pack()                       # canvasの配置確定
    points = circle.circle()
    circle.display(canvas, points)
    display(canvas, points)                 # 描画関数 (display) の呼出
    root.mainloop()                     # ルートフレームの実行ループ開始

if __name__ == '__main__':              # 起動の確認 (コマンドラインからの起動)
  main()                            # main関数の呼出
```

○実行コマンド

```
MBP:Chap02 hayashikippei$ python3 diamond.py
# of points -> 40
```

○実行結果

???



○考察

直線によるカージオイドの描画のプログラムを参考にした。circleモジュールを利用して円周上の座標値を得て、各頂点に対して、自分自身と隣の２点を除いた 多角形の全ての頂点間と直線を引くfor文を書いた。len(points)の余りを取ることで１周を超えたら調整するようにした。

□課題2.7 - 2.2節 章末課題:直線によるネフロイドの描画

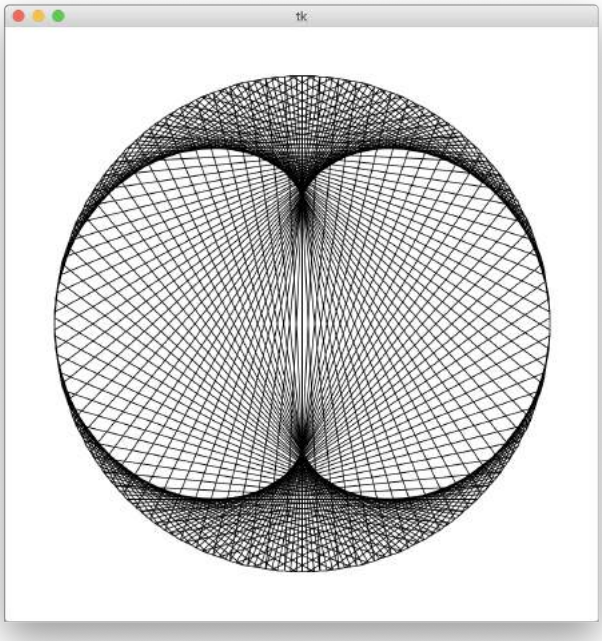
○プログラムリスト

```
from tkinter import *
import circle

W, H = (600, 600)

def display(canvas, points):
    for i in range(len(points)):
        j = (3 *i) % len(points)
        canvas.create_line(points[i], points[j])

def main():
    root = Tk()                         # ルートフレームの作成
    canvas = Canvas(root, width = W, height = H, highlightthickness=0)
                                # canvasの作成
    canvas.pack()                       # canvasの配置確定
    points = circle.circle()
    circle.display(canvas, points)
    display(canvas, points)                     # 描画関数 (display) の呼出
    root.mainloop()                     # ルートフレームの実行ループ開始

if __name__ == '__main__':              # 起動の確認 (コマンドラインからの起動)
  main()                            # main関数の呼出
```

○実行コマンド

```
MBP:Chap02 hayashikippei$ python3 nephroidLine.py
# of points -> 256
```

○実行結果



○考察

直線によるカージオイドの描画を参考にし,circleモジュールで得た正n角形の各頂点PiからP2iに対角線を引いた。

□課題2.7 - 2.2節 章末課題:円によるネフロイドの描画

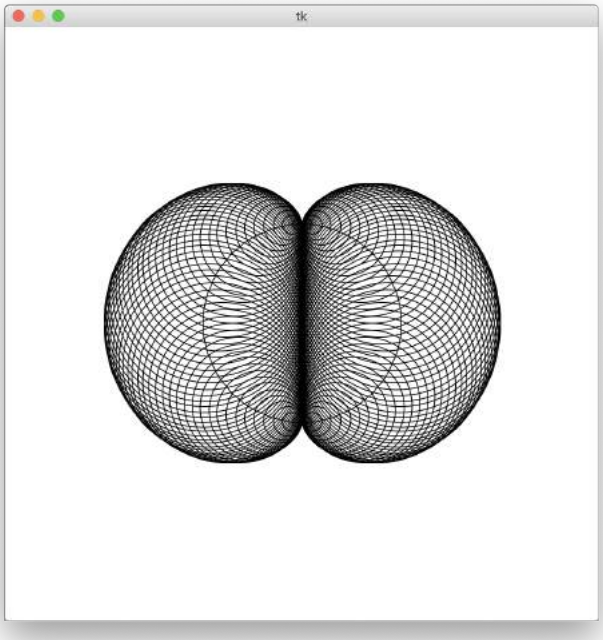
○プログラムリスト

```
from tkinter import *
import circle

W, H = (600, 600)

def display(canvas, points):
    for i in range(len(points)):
        r = abs(points[i][0] - 300)
        ul = (points[i][0] - r, points[i][1] - r)
        lr = (points[i][0] + r, points[i][1] + r)
        canvas.create_oval(ul,lr)



def main():
    root = Tk()                         # ルートフレームの作成
    canvas = Canvas(root, width = W, height = H, highlightthickness=0)
                                # canvasの作成
    canvas.pack()                       # canvasの配置確定
    points = circle.circle((300, 299),100)
    circle.display(canvas, points)
    display(canvas, points)                 # 描画関数 (display) の呼出
    root.mainloop()                     # ルートフレームの実行ループ開始

if __name__ == '__main__':              # 起動の確認 (コマンドラインからの起動)
  main()                            # main関数の呼出
```

○実行コマンド

```
MBP:Chap02 hayashikippei$ python3 nephroidOval.py
# of points -> 128
```

○実行結果



○考察

円によるカージオイドのプログラムを参考にした。具体的に、書き換えたのはr = abs(points[i][0] - 300)の 1 行だけである。 基円の中心(300,299)を通る直線をx = 300としたため、各頂点において描く円はその頂点を中心とし、基線に接する円。この場合半径はabs(points[i][0] - 300) となる。画像を見ても、正しく描画できていることが確認できる。

□課題や授業に関して

○レポート作成に要した時間

4.0時間

○特に苦労した点

なし。強いて言えば例題の問題数が多かった。

○授業についての感想や希望

例題の考察、何を書けば良いかよくわかりません。