Simple Polygonize

Demonstration of algorithm for finding a simple polygon that passes through given points. Implemented for CSN-523 (Computational Geometry) course assignment.

Algorithm

1. Choose a suitable reference point (in the code, used arithmetic mean of each axis)
2. Calculate angle of line segment joining each given point to the reference point, with respect to the X axis
3. Sort the points based upon the angle of incidence calculated above
4. Join adjacent points in the sorted order using line segments
5. Join first and last point with a line segment

Code

```
import sys
import numpy as np
import cv2

SIZE = 512

def gen_rand_points(N):
    from random import randint

    points = [(randint(0, SIZE - 1), randint(0, SIZE - 1))
              for _ in xrange(N)]

    return points

def order_as_polygon(points):

    refX = int(sum(x for x,y in points) / float(len(points)))
    refY = int(sum(y for x,y in points) / float(len(points)))

    def angle(point):
```

```
        x,y = point
        relX,relY = x - refX, y - refY
        return np.arctan2(relX, relY) # takes care of quadrant calculation

    return (refX, refY), sorted(points, key=angle)

def plot(points, ref):
    img = np.zeros((SIZE,SIZE,3), np.uint8)

    lines = zip(points[:-1], points[1:]) + [(points[-1], points[0])]

    for p1, p2 in lines:
        cv2.line(img, p1, p2, (255, 255, 255))

    font = cv2.FONT_HERSHEY_SIMPLEX

    for i, p in enumerate(points):
        cv2.circle(img, p, 3, (0, 0, 255), -1)
        cv2.putText(img,str(i),p,font,0.5,(255,255,0),2)

    cv2.circle(img, ref, 3, (0,255,0), -1)

    return img

def display(img):
    cv2.imshow('simple polygonize', img)
    return chr(cv2.waitKey(0) & 0xff)

def main():
    try:
        N = int(sys.argv[1])
    except:
        N = 3

    def do_new():
        points = gen_rand_points(N)
        ref, points = order_as_polygon(points)
        return plot(points, ref)

    img = np.zeros((SIZE,SIZE,3), np.uint8)
    cv2.putText(img, "Simple Polygonize", (130, 130),
```

```
                cv2.FONT_HERSHEY_SIMPLEX, 1, (256, 256, 256), 2)
    cv2.putText(img, "Help", (230, 230),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (256, 256, 256), 2)
    cv2.putText(img, "+/- change N", (190, 260),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (256, 256, 256), 1)
    cv2.putText(img, " r    randomize points", (190, 280),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (256, 256, 256), 1)
    cv2.putText(img, " q    quit", (190, 300),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (256, 256, 256), 1)
    next = display(img)
    while True:
        if next == '+':
            N += 1
        elif next == '-' and N > 3:
            N -= 1
        elif next == 'q':
            break
        elif next == 'r':
            pass
        else:
            next = display(img)
            continue
        img = do_new()
        cv2.putText(img, "N = %d" % N, (5, SIZE - 5),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (127,127,127), 1)
        next = display(img)
    cv2.destroyAllWindows()

if __name__ == '__main__':
    main()
```

License

All parts of the code are covered under the MIT License

How to Run

Simply run the `simply-polygonize.py` file using `python simply-polygonize.py` or `python simply-polygonize.py {N}`, where `N` is the number of points required.

When the program runs, it will prompt with the help text:

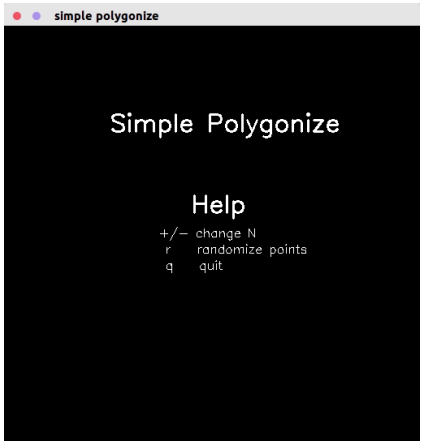
```
+/- Change N
 r  randomize points
 q  quit
```

Press any of the above keys to do the corresponding action.

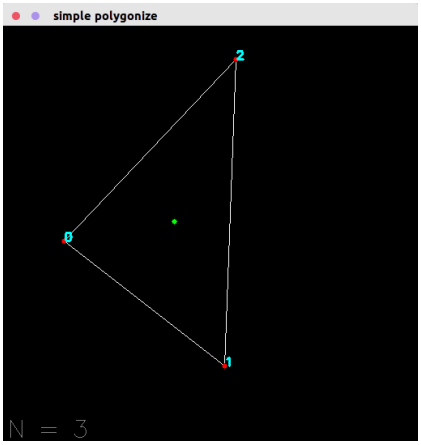
Demonstration

A video of the code in action, is available at
https://www.youtube.com/watch?v=WXUb6b_7CIk

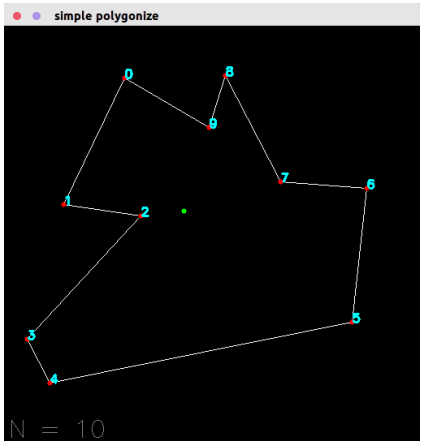
Some screenshots are shown below:



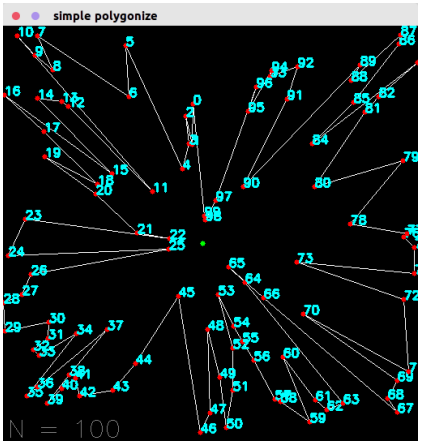
(a) Intro Screen



(b) N=3



(c) N=10



(d) N=100