Webology (ISSN: 1735-188X)
Volume 15, Number 1, 2018

Two Algorithms For Constructing A Regular Polygon Of Given Side Length

Sajad A. Sheikh⋆

⋆Department of Mathematics, University of Kashmir, South Campus, Anantnag 192101, Jammu and Kashmir, India.

Abstract. In this paper we propose two algorithms for the construction of regular polygons. The first algorithm is for generating a regular polygon with the coordinates of vertices of one side given and the second algorithm generates a regular polygon with a given side length using properties of nth root of unity. Both algorithms are of computational complexity O(n). The algorithms not only generate the requisite polygon but enables us to compute the exact coordinates of the vertices, as they may be required in certain problems in computational geometry.

Keywords: Regular Polygon; Algorithm; Graphics; Geometry; Python.

1 Introduction

As a result of ever increasing power and capability of graphics hardware, computer graphics algorithms are being increasingly used in many scientific and technological areas [1]. Obviously the generation of basic geometric shapes such as circles, lines, triangles, polygons is of fundamental significance in the field. Polygons in particular are widely put to use in computer graphics for the rendering of three dimensional objects. Usually triangular, polygons arise when an object's surface is modelled, vertices are selected, and the object is rendered in a wire frame model. As wire frame models are quicker to display, it is only natural that polygons should arise as stage in computer animation. The polygon count refers to the number of polygons being rendered per frame. As the fifth generation of video game consoles became commercially widespread, the use of polygons became more common, and with each succeeding generation, polygonal models became increasingly complex [2, 3]. In this article we propose two algorithms for the generation of regular convex polygons with any number of sides. Both algorithms have a time complexity of O(n). The proposed algorithms also enable to compute the exact coordinates of the vertices, which are needed in some problems arising in computational geometry.

3. First Algorithm for constructing a regular Polygon

In this section we are going to present our first algorithm for the construction of a regular polygon with one side given. This algorithm assumes that the coordinates of the two vertices of a side are known.

Consider a segment OA on the X-axis, where O is the origin of XY-plane. A simple observation tells us that a regular polygon with one side as OA can be obtained by the following procedure (see Figure 2):

1 Take $\theta = \frac{(n-2)\pi}{n}$.

2 Rotate OA about A in clockwise direction about θ, taking point O to O'.

3 $A \to O$ and $O' \to O$ and repeat step 2 till we reach back to origin. In total we need $(n-1)$ steps to obtain a regular polygon of n sides. So the procedure essentially involves rotating a segment, AB in XY-plane with the coordinates of A and B known, and keeping track of new positions of B. We find out the transformation that achieves precisely this objective as follows:

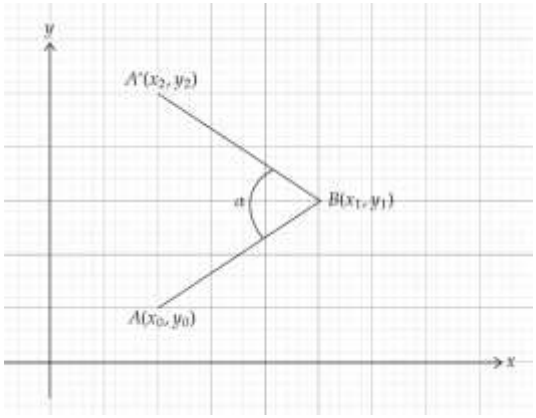

Figure 1: Rotation of AB with angle α.

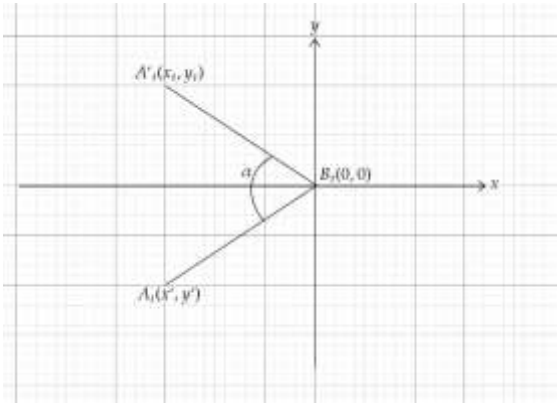

Figure 2: Translation of axes.

Consider a segment AB with $A \equiv (x_0, y_0)$ and $B \equiv (x_1, y_1)$, and rotating AB around B clockwise by angle α. Shift origin to B so that the new axes are X' and Y'. The coordinates (x', y') of A relative to $X' - Y'$ axes are given by

$$x_0 = x' + x_1,$$
$$y_0 = y' + y_1$$

or equivalently, $x' = x_0 + x_1, y' = y_0 - y_1$.
In complex form A in $X' - Y'$ is given by

$$x' + iy' = (x_0 + x_1) + i(y_0 + y_1)$$

Since multiplying a complex number z by $e^{i\alpha}$ rotates counter clockwise by an angle α, the coordinates of A_t are obtained as

$$\begin{aligned}
\overrightarrow{O'A'_t} &= (x' + iy') \cdot e^{-i\alpha} \\
&= (x' + iy')(\cos\ \alpha - i\sin\ \alpha) \\
&= (x'\cos\ \alpha - y'\sin\ \alpha) + i(y'\cos\ \alpha - x'\sin\ \alpha) \\
&= \begin{bmatrix} \cos\ \alpha & \sin\ \alpha \\ -\sin\ \alpha & \cos\ \alpha \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \\
&= \begin{bmatrix} \cos\ \alpha & \sin\ \alpha \\ -\sin\ \alpha & \cos\ \alpha \end{bmatrix} \begin{bmatrix} x_0 - x_1 \\ y_0 - y_1 \end{bmatrix},
\end{aligned}$$

which yields the following values of A'_t.

$$\begin{aligned}
x_t &= (x_0 - x_1)\cos\ \alpha + (y_0 - y_1)\sin\ \alpha, \\
\text{and } y_t &= -(x_0 - x_1)\sin\ \alpha + (y_0 - y_1)\cos\ \alpha.
\end{aligned}$$

Translating origin back to $B(x_1, y_1)$, the coordinates of new A_t with respect to the coordinate system prior to both translation and rotation are

$$\begin{aligned}
x_2 &= (x_0 - x_1)\cos\ \alpha + (y_0 - y_1)\sin\ \alpha + x_1, \\
\text{and } y_2 &= -(x_0 - x_1)\sin\ \alpha + (y_0 - y_1)\cos\ \alpha + y_1.
\end{aligned}$$

If $(x_i, y_i), i = 0,1,2, \ldots n$ are the coordinates of the vertices in anticlockwise order, then at the i^{th} iteration;

$$\begin{aligned}
x_{i+1} &= (x_{i-1} - x_i)\cos\ \alpha + (y_{i-1} - y_i)\sin\ \alpha + x_i \\
y_{i+1} &= (x_{i-1} - x_i)\sin\ \alpha + (y_{i-1} - y_i)\cos\ \alpha + y_i
\end{aligned}$$

or,

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\ \alpha & \sin\ \alpha \\ -\sin\ \alpha & \cos\ \alpha \end{bmatrix} \begin{bmatrix} x_{i-1} - x_i \\ y_{i-1} - y_i \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Illustration of the above algorithm: Let's try to construct a square with two vertices at $(0,0)$ and $(1,0)$ using first algorithm.

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

For $\alpha = 90^\circ$

$$\begin{aligned} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

which is the same as the starting vertices.

3. Second Algorithm for constructing a regular Polygon

In this section we present our second algorithm for the construction of a regular polygon with given side length. This algorithm utilises the geometrical properties of the roots of unity for the construction of a regular polygon.

Firstly, we start with the fact that the nth roots of unity are given by:

$$\begin{aligned} x^n &= 1 \\ \text{or } x^n &= e^{2\pi k\iota} \end{aligned}$$

Some basic algebra reveals that the n roots can be written as

$$\omega_k = e^{\frac{2\pi k}{n}\iota}, k = 0,1,2, \cdots, n-1$$

It can easily be seen that all the n complex numbers given by equation (3.1) lie on the circle

$$|z| = 1$$

Further,

$$\left|\omega_j - \omega_k\right| = \text{cis } \frac{2\pi j}{n} - \text{cis } \frac{2\pi k}{n},$$

where $\text{cis}\ \theta = \cos\ \theta + \iota \sin\ \theta$. When ω_i, ω_j are adjacent vertices of a polygon then $k = j + 1$, so that the above expression becomes

$$\left|\omega_j - \omega_k\right| = 2\left|\text{cis}\ \frac{j\pi}{n} - \text{cis}\frac{(j+1)\pi}{n}\right|$$

$$\left|\omega_j - \omega_k\right| = 2\cos\ \frac{2\pi}{n}$$

In order to show that $\left|\omega_j - \omega_{j+1}\right|$ is indeed constant for a given n, we use another property of the n^{th} roots, which is $\omega_j = \omega^j$, so that

$$\left|\omega_j - \omega_{j+1}\right| = \left|\omega^j(1-\omega)\right|$$

$$\left|\omega_j - \omega_{j+1}\right| = |1-\omega|$$

with $\omega = \cos\ \frac{2\pi}{n} + \iota \sin\ \frac{2\pi}{n}$, the distance $l(n)$ between adjacent roots $|1-\omega|$ evaluates to

$$l(n) = \sqrt{\left(1 - \cos\ \frac{2\pi}{n}\right)^2 + \left(\sin\ \frac{2\pi}{n}\right)^2}$$

Analogously, the distance between two consecutive nth roots of a positive real number r is given by

$$l_r(n) = r^{\frac{1}{n}}\sqrt{\left(1 - \cos\ \frac{2\pi}{n}\right)^2 + \left(\sin\ \frac{2\pi}{n}\right)^2}$$

This means if the length of regular polygon with n sides is to be l, the vertices will be given by the n^{th} primitive roots of the real number r where, r is given by

$$r = \left(\frac{l}{\sqrt{\left(1 - \cos\ \frac{2\pi}{n}\right)^2 + \left(\sin\ \frac{2\pi}{n}\right)^2}}\right)^n$$

Hence the steps in our second algorithm can be summed up as follows

1 For given number of sides n and side length l we take

$$r = \left(\frac{l}{\sqrt{\left(1 - \cos\frac{2\pi}{n}\right)^2 + \left(\sin\frac{2\pi}{n}\right)^2}} \right)^n$$

2 For $k = 0,1, \dots, n-1$

$$x_k = r^{1/n}\cos\left(\frac{2\pi k}{n}\right)$$
$$y_k = r^{1/n}\sin\left(\frac{2\pi k}{n}\right)$$

3 The output is the ordered list of vertices $[(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$ of the required polygon in counter clockwise fashion.

References

[1] M. Ramakrishnan, Advanced Methods in Computer Graphics with examples in OpenGl, Springer,2012.

[2] S. Rich, A brief History of Video Games.

[3] S. Daniel, Practical Geometry Algorithms with C++.

Python Code for Implementation

```
import sympy
from sympy import symbols
from sympy import *
init_printing(use_unicode=True)
x0, y0, x1, y1, x2, y2, t= symbols('x0 y0 x1 y1 x2 y2 t')
x0, y0, x1, y1 = 0, 0, 1, 0
n = int(input("enter the number of sides"))
t = (n-2)*pi/n
cord = [] #stores the co-ordinates of vertices

for i in range(n+1):
  x2 = cos(t)*(x0 - x1)+sin(t)*(y0 - y1)+x1
  y2 = -sin(t)*(x0 - x1)+cos(t)*(y0 - y1)+y1
  x2 = simplify(x2)
  y2 = simplify(y2)
  cord.append(Matrix([[x2],[y2]]))
  x0,y0 = x1,y1
```

```
  x1,y1 = x2,y2

import matplotlib.pyplot as plt
xlist=[]
ylist=[]
for i in range(n+1):
  xlist.append(cord[i][0])
  ylist.append(cord[i][1])
ax = plt.axes()
ax.set_aspect(1)
plt.plot(xlist, ylist)
plt.show()
```