Project 1: Guarding an Art Gallery

By: Sunshine Schneider and Sarah Branch

Description:

Using code to triangulate a polygon (see previous project), 3-color the vertices. Highlight the floor(n/3) guards.

Source Code:

The code used in polygon_triangulate.py is https://people.sc.fsu.edu/~jburkardt/py_src/polygon_triangulate/polygon_triangulate.html
This code makes the polygon into triangles and needs numpy and you need to input the vertices counter-clockwise.

What is the 3 color proof/Fisk proof?

The Fisk three color proof is that when there are n vertices and it is broken into n-2 triangles (every polygon has a triangulation and has n-2 triangles). You color each vertice a different one of three colors, so no triangle repeats one of the same colors. You can then put a guard at the color covering the least number of vertices which will always be sufficient, but not necesarry for covering the interior of the polygon.

What does the code do?

The code triangulates a polygon and numbers the vertices. The vertices of the triangles are then colored one of three colors, which in this case are represented by numbers 0, 1, and 2. No triangle can have the same color so each vertice will be different. This is done through a greedy color algorithm which assigns each vertex the first available color. It then returns the amount of guards needed to be sufficient to cover the interior of the polygon. It follows the proof that floor(n/3) vertices are sufficient to cover the interior of a polygon.

Process:

- Finding source code to triangulate polygon
- Using greedy color algorithm to color each vertex

Functions!

polygon_triangulate() → triangulates polygon.

tricolorize() function → assigned each vertex to one of 3 colors
If there is no vertice connected with 0, it assigns 0 and so one

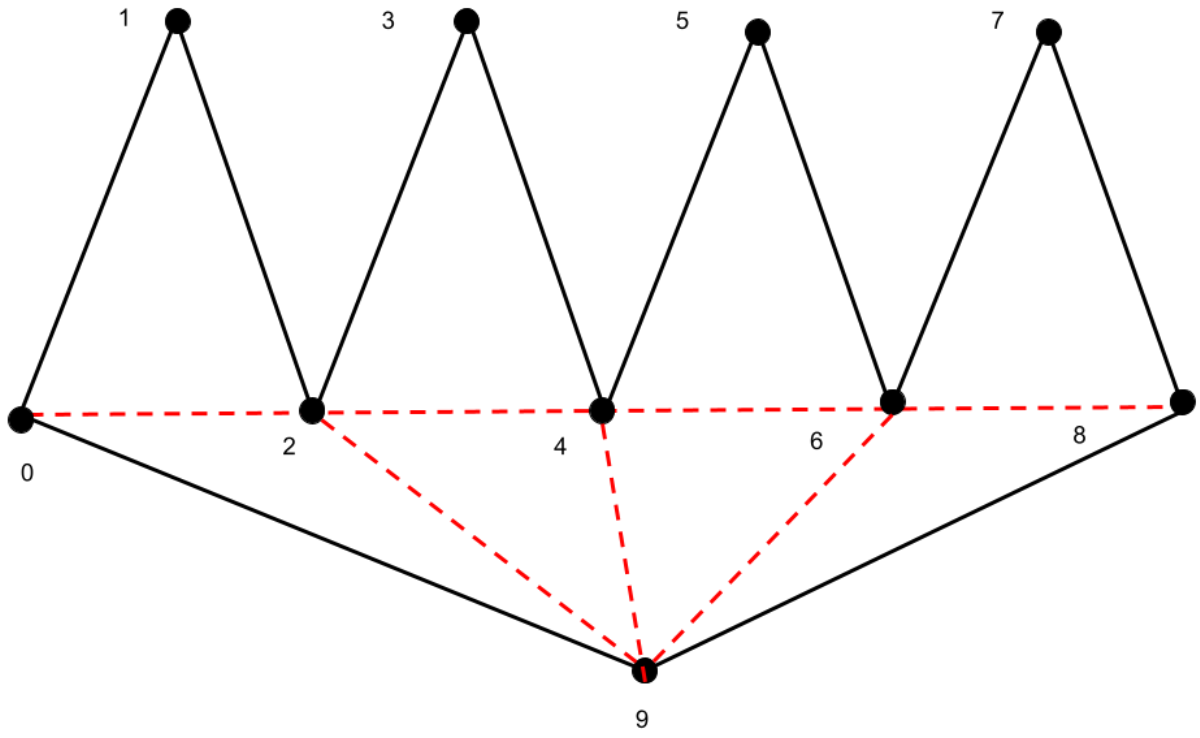
0, 1, 2 are the colors

coloredtriangle() → takes all the color assignments for each triangle and it's vertices and puts it in a nested list

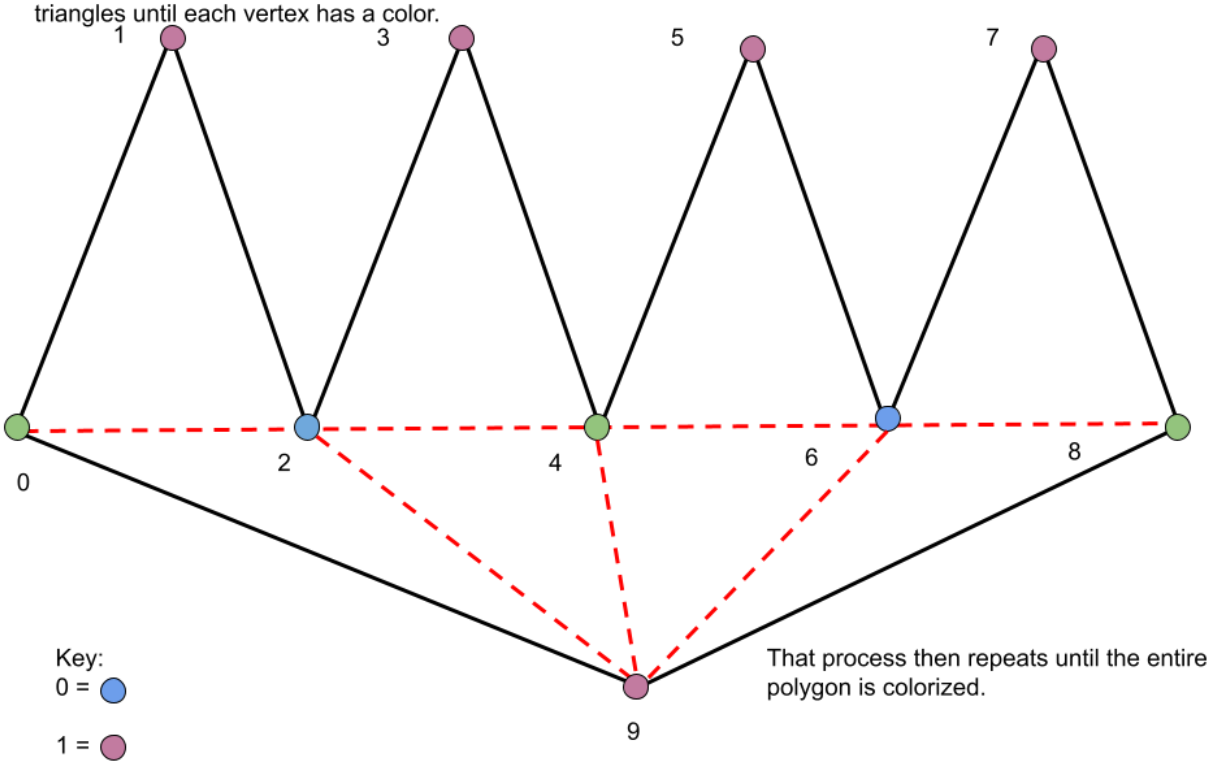
minguards() → checks for the minimum number of guards and returns that value by seeing which color number is used the least.

Drawing Example for a n = 10 Polygon:

The polygon_triangulate.py file triangulates the polygon and assigns a vertice number to each vertex in a clockwise motion while require the x- and y- vertices to be inputted in counter-clockwise using the polygon_triangulate(n, x, y) function.



The tricolorize(n, x, y) function checks each vertex, starting at triangle 6, 9, 4 assigning different colors to each point. It then moves to triangle 4, 9, 2. Since point 4 is already assigned the color 2, and point 9 is assigned the color 1, point 2 is assigned the color 0. This process then repeats for all triangles until each vertex has a color.

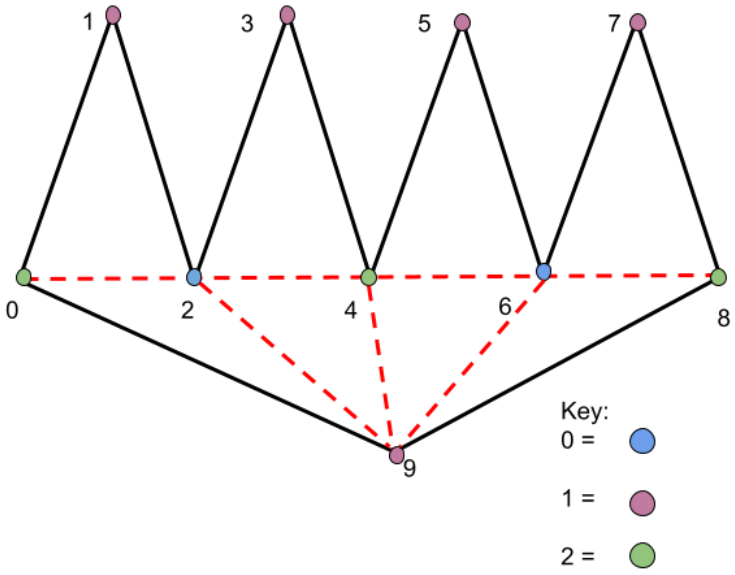


Key:
0 =
1 =
2 =

That process then repeats until the entire polygon is colorized.

The minguards(n, x, y) function finds the minimum number of guards by seeing which colored value is used the least and returns the number of guards that will always be sufficient to cover the interior of the polygon.

Here, minguards(n, x, y) would print out 2 since the color 0 is used the least amount of times at only 2 times. It also follows floor(n/3) which is in this case is floor(10/3) = 3 saying 3 guards will be sufficient but not needed for all polygons. In the case of out polygon with n = 10, 3 guards are not needed since 2 guards are can cover the interior so the floor(n/3) theorem is still true and applies here.



Through the image below, we see that both guards can see the entire interior of the polygon showing that 3 guards would be sufficient to see the whole interior, but only 2 guards are necessary.