Steps of understanding

1. Convex polygons only require 1 guard, there’s nowhere to hide
    a. Demonstrate with a triangle
2. Some shapes are harder to guard than others, sharp corners and hidden nooks/crannies require more guards
    a. Let the user position guards in “U” gallery
    b. Let the user position guards in the “spiral” gallery
    c. Let the user position guards in “comb” gallery
3. What if we cut the gallery into simpler shapes?
    a. Walk the user through triangle decomposition of an arbitrary, complex polygon
    b. Walk the user through triangle decomposition of the “comb” gallery
4. Now, we just need to make sure that each sub-gallery is guarded.
    a. Show that the triangular decomposition is a planar graph
    b. Reference proof that all planar graphs are 3-colorable (This should be its own lesson)
5. Now, the gallery has been broken into triangular sub-galleries that are easily guarded from any vertex. By 3-coloring the vertices, we can place a guard on any color and be sure that every triangle will be guarded.

Convex polygons only require 1 guard

The user will be given a triangle with a blue circle representing a guard in the center. The user can click anywhere in the triangle to move the guard. No matter where the user moves the guard, it will draw lines to the vertices of the triangle. A paragraph above will explain that if a guard can see two adjacent vertices, the guard must be able to see the entire edge between them. Since the guard can always see all three vertices of the gallery, he must always be able to see the whole perimeter.

Some shapes are harder to guard than others

The user will be given a set of buttons to navigate between different galleries. The galleries range from a simple U that shows sometimes we need more than one guard to a spiral gallery that will need many guards to see all the twists to the “worst case” comb gallery that has lots of spikes which aren’t visible to each other.

In each gallery, the user can place guards by clicking within the polygon or move guards by clicking and dragging an existing one around the polygon. If the user drags a guard outside of the polygon, it will be deleted. Instead of the guards drawing lines to visible vertices, the guards will light up edges that they can fully see.

What if we cut the gallery into smaller shapes?

The user will be given a set of buttons to navigate between the galleries from the previous section. In each gallery, the user can click on vertices of the gallery to cut them out of the gallery, forming a triangle. Not all vertices can be cut without destroying the gallery, so the user will be prompted with an error message if they try to cut a vertex destructively. That vertex will also turn red until the user successfully cuts another vertex. Uncut vertices display as blue circles and cut vertices turn green.

3-Coloring a planar graph

The user will be given a set of buttons to navigate between the galleries from the previous section, after they've been triangle decomposed. In each gallery, the vertices all start blue and the user can click any vertex to cycle it through blue -> red -> green -> blue.

3-coloring planar graphs could be an entire lecture/lesson to itself. I think that we have to assume some knowledge of vertex cover and graph coloring.

Putting the pieces together

Show the user that there is a vertex of each color on every triangle. This means that you could place a guard on every blue (or red or green) vertex and guard every triangle. This puts an upper bound on the solution to the art gallery problem of n/3 where n is the number of vertices in the gallery.

Running my sample

I have written javascript demos and html webpages for the first three sections above. I have not completed the 3-coloring page. You can run server.py, just a simple http server and open localhost:8000 to view my demo.