CMSI 371-01
COMPUTER GRAPHICS
Spring 2015

Assignment 0226 Feedback

Outcomes that ultimately cover both 2D and 3D max out at | for now because we are dealing only in 2D. They will expand to their full potential with the 3D course work.

Daniel Levine

dannymlevine / dannymlevine@gmail.com

Your filters are functionally right, but (a) the single-pixel filters both look like variations on the same theme, and (b) your neighborhood filters don't look like they actually use the neighborhood. Of course this is just based on visual inspection; we'll see what the code says. For your gradient, the radial effect looks right but takes much longer than necessary. I suspect some efficiencies lurking in the code.

1. Watch the copied comments! This one certainly does not apply anymore, and should be deleted. (*4c*)
2. You certainly made a few filters, but they were supposed to be added to the `Nanoshop` object as "library" functions. More of a software structure than a graphics issue, but still off-spec from what was instructed in the assignment. (*4a*, *4b*)
3. As suspected, your single-pixel filters are more or less the same thing: do at most a single arithmetic operation to each color component. These are all technically right, but it didn't stretch you very far creatively. (*2c*, *3c*)
4. And here, also as suspected just upon execution, one of your "neighborhood" filters are ultimately no different from your single-pixel filters, meaning that it doesn't make use of the neighborhood! The point of doing this set of filters with the new `applyFilter` function was missed on this one. (*2c*, *3c*)
5. Gack, your code presentation took a nosedive here! (*4c*)
6. *** Note that this is an unnecessary calculation—the given `x` and `y` are *always* the same distance from the center of the circle! A better change is to actually add a `radius` parameter to `plotCirclePoints`, which the caller then simply supplies.

   But wait—there's more! *You are not actually calculating the radius here.* Note that `x` and `y` are already *offsets* from the center (`xc`, `yc`)…meaning that *they are already the distances from the center*. Subtracting them from `xc` and `yc` gives you a false radius (go ahead, `console.log(radius)` with your code). What you really wanted to calculate is `Math.sqrt(x*x + y*y)`—but, as mentioned in the first paragraph of this note, that computation is ultimately not necessary. (*4b*)
7. *** The unexpected consequence that the bug in note #6 triggers is that it hides a deeper flaw in the circle-filling logic. What your code is clearly doing here is drawing concentric circles. Because the code calculates the wrong radius, *it is actually drawing concentric circles at fractional distances*. This has the dual [undesirable] effect of (a) making your circles look completely filled even though the approach to filling them isn't right, and (b) making it take noticeably longer to fill your circles than an optimal implementation of this algorithm. This is why the latency didn't feel right to me—I know how the correct implementation performs and it is definitely much faster than this one. This bug is the reason why.

   With respect to (b) there *is* a way to guarantee full coverage, but that requires breaking out a little bit from the notion that you are drawing a circle. Instead, you should think about this as a generic fill operation. Note that the radial gradient will still be computable in that scenario. (*2d*, *4a*)

1a — |

2c (max |) — / …Overall, you pretty much implemented just one filter, with slightly varying values. Even the sole neighborhood one is based on the same concept, except that you summed things up first.

CMSI 371-01
COMPUTER GRAPHICS
Spring 2015

Assignment 0226 Feedback

Outcomes that ultimately cover both 2D and 3D max out at | for now because we are dealing only in 2D. They will expand to their full potential with the 3D course work.

2d — / …Yes, this is mainly the seemingly-correct-but-ultimately-quite-wrong radial gradient implementation. The core pitfall here was missing that `x` and `y` are already distances to the center. That created a domino effect that produced very misleading results, which in turn hid an even deeper misconception about how to correctly fill a circle. I think this instance is worth talking through in case you're not fully clear on what's going on with the code; let me know if you want to chat about it.

3c — / …And this one is for the same overall reasons as *2c*.

4a — / …Between the single general filter style, the missed neighborhood computation, and the sneaky radial gradient bug, we have some notable areas of improvement in functionality here.

4b — | …Despite the missing functionality, the code itself was fairly clean. The slight ding is for the misunderstanding about the radius plus being off-spec in `Nanoshop`.

4c — | …You know it was generally going well until `plotCirclePoints`. But that one just shattered things. Fortunately the rest of the code presentation was decent, minimizing the impact.

4d — | …I think you could have used some web browsing to get ideas for more filter types.

4e — + …This is borderline, really—but I can also see the work being phased the way you did, with the radial gradient being a single commit.

4f — +