OpenCV Project

TL;DR: Write me a script that demonstrates that you understand the material from the book at more than just a surface level.

What does this actually mean/what will I be looking for?

- Before you start, clone the Github Classroom repository so I can see your work. Make sure that you push.sync regularly.
- There are 9 chapters of material in the textbook (so, ignoring the first two), plus the extra 'chapter' on morphology. You should have at least one idea from each of 6 chapters represented* (see next bullet point). For example, Chapter 3 will be in everyone's, because you will load and display images.
- Make a checklist saying which Chapters are represented, and which functions you used that represent those Chapters. Put this in your README.md.
- If you want to use functions from OpenCV that were not covered in the text, or from any other Computer Vision technique, you can remove one or more chapters from your list. Check with me, but the idea I have in my head is basically one function properly used that we haven't studied can replace one Chapter from the text.
- Do not use any of the images you used in your first pass through the text. Find some new (school-appropriate) images: Use photos you've taken. Take pictures of your dog. Download some stuff from the web. Take screenshots playing your favorite video game.
- Don't try to swallow all the Chapters in one action. It can/should be a series of actions so that we can see what's happening as it's happening.
- I will assume that I should run the program from command-line, and any command-line arguments will be described for me using `argparse help` statements.
- I will assume, once I've started the program, that I should just hit the spacebar each time something displays to get to the next thing. If not, your program will have to tell me what to do to get to the next step.
- You must do some things in your program that are not exactly what's in the book, for example: Mess around with the parameters. Use different shapes. Use shapes in different ways.
- YOUR SUBMISSION CHECKLIST:
    - Make sure your work is committed to Git and pushed/synced to Github.
    - Make sure you have a checklist in your README.md that shows what Chapters you used.

Presentation

You will have to present your work to the rest of the class. This is a **low-key** affair, don't stress about this. Do **not** make a slideshow. Do **not** prepare a fancy presentation. What I'm looking for is for you to bring your laptop up to the front of the room, connect to the projector, and run your

code. As you step through it, talk a little bit about what you did and what your code does and what you were thinking. If there's something interesting/different in the code, you can show us that. Then, I'll open the floor for questions, and I'll probably harass you with a question or two. My favorite question to ask is: “What was hard about what you did?” Again, this is a low-key presentation. How long is it? As long as it needs to be. Probably 3 minutes-ish. If you have a lot to talk about, maybe 5 minutes.

The goal of the presentation is for other people to see the cool stuff you are working on. I find that students in this class are always working on interesting things, and it is a shame to have students do all that cool stuff and have almost no one see it. So, you'll have a dozen or so other students see what you are doing.

The bottom line

So, do cool stuff and show it off. Have fun with it.

OK, not as much fun, but there’s a Rubric. Here’s the rubric breakdown. Notice that **Full Credit is the “Meets Expectations” column**. If you go into the “Exceeds Expectations” column on any row, that’s a +10%, up to a maximum of 100%.

1. Checklist: 50%
2. Independence and Intellectual Curiosity: 10%
3. Functioning Code: 10%
4. Coding Practices: 10%
5. Design: 0%
6. Ambition: 10%

Notice: This adds up to 90%. To get above a 90%, you will need to go into “Exceeds Expectations” in one row, or convince me that you have hit “Meets Expectations” in the Design row.