Using Robotics To Make Blinds Smarter

Maryo Botros
mbotros@oxy.edu
Occidental College

1 Introduction

This document serves describes the Oxy CS Comps process. The first section describes the timeline of comps, as well as the main requirements of a comps project. Since one of the main deliverable is a final paper, there is then an interlude on LaTeX, which you will use to write the paper. This document then goes into the sections of that final paper and what is expected of it. We conclude with some miscellaneous tips for successfully completing comps.

2 Computer Science Comps

Occidental's "Comprehensive Requirement" is a graduation requirement that serves two goals: "to provide an opportunity for senior students to synthesize the essential[s] [...] of their academic field", and "to provide an opportunity for students to demonstrate competence in their field" [6]. For computer science, the breadth of the field means that comps leans heavily towards the second goal: most CS comps are large coding projects that showcase what a student has learned in their four years at Oxy. In that sense, a CS comps project is more like a Studio Arts comps project: you could draw from multiple subfields or focus on just one, but the essence is to take on a complex project of your choice while situating it within the methods of CS.

2.1 Timeline

Since comps is a significant undertaking, the CS major curriculum has two courses to support comps projects. Most students will take CS Junior Seminar in the spring of their junior year, then take CS Senior Seminar in the fall of their senior year. One of the main goals of Junior Seminar is to prepare students for their comps project, including selecting a topic, exploring the literature, and writing a detailed proposal of what they will do. The entirety of Senior Seminar is dedicated to students completing their comps; however, it is necessary to separate the Senior Seminar course from the comps graduation requirement. Senior Seminar is a required course for the CS major, and it concludes at the end of the fall semester with a grade. Comps, on the other hand, is an Oxy graduation requirement, and concludes at the end of the senior year with either "fail", "pass", or "pass with distinction" [6]. While the sole purpose of Senior Seminar is to support students working on comps, it is possible for a student to complete the *course* without completing the *graduation requirement*. This could occur, for example, if the student ran out of time to finish the project, but submitted all the assignments for Senior Seminar. In that case, the student will continue working in the spring senior semester to complete their comps, until they have a "pass" or "pass with distinction" result. A summary of this timeline can be found in Table 1.

Semester	Course	Deliverable(s)
Junior Spring	COMP 390	Comps proposal; Tutorial report
Summer	-	Additional research
Senior Fall	COMP 490	Comps final paper; Code repo
Senior Spring	-	Revisions; Honors

Table 1: Timeline summary for comps.

2.2 Deliverables

The two main deliverables for CS comps is the code from the project, submitted together with its documentation as a GitHub link, and a paper about the project. While the code component is straightforward and uncontroversial, some students may chafe against the idea of a paper. Contrary to popular believe, the job of a software engineer does not only involve writing code. It also requires a large amount of written communication, such as tickets for bugs reports, proposals for new features and why it will improve the product, comments on why a class is necessary or how a function works, and documentation of the system architecture. As software engineers rise from junior to senior positions, these responsibilities only increase. It is often lamented that not enough engineers know how to write [5, 1], which is why Google has created courses on technical writing [2], and "soft skills" are routinely touted as necessary for software developers [3].

Given these goals, the final paper should be written for an advanced CS undergraduate: someone who has finished Math Foundations, Data Structures, and Computer Organization, but may not have taken the appropriate upper-level courses. The paper does not need to explain everything, but

should provide enough relevant details so the average reader can understand your results; a walkthrough of the sections of the paper can be found in Section 4. The paper will be written in LaTeX, to make all final papers uniform in style and to allow for psuedocode and mathematical equations; a template will be provided and required. To ensure there is sufficient context and detail, the paper should be about seven pages in length (excluding figures, tables, code, and references). The target audience for the paper is other computer science majors in their final year of college, but who may not have taken the same electives that you have. That is, you should assume that they have working knowledge of data structures, discrete math, and computer systems, but not necessarily any knowledge of artificial intelligence, machine learning, databases, or other more advanced topics.

The other major deliverable is the code, which should be made available on a public GitHub repository. Accompanying the code should be two pieces of documentation, both included in a `README.md`. First, there should be instructions for how to run the code, which should including how to find and install necessary software and packages, how to find and download relevant datasets, and what other requirements (hardware, spatial needs for VR, etc.) are needed. This should be written for an audience who is interested in reproducing your work and verifying your results. Second, there should be a brief overview of how the code is organized, including the major components that exist and the content and format of the data that is passed between them. This should be written for someone who is interested in extending your project — say, by tweaking the parameters, or adding new functionality, or applying it to a new context.

More details about both deliverables will be provided in the Junior and Seminar Seminars.

2.3 Selecting a Comps Project

For many students, selecting a topics for comps is a major source of stress. While this is an important decision, it is also one for which you will receive a lot of support. Choosing a topic, together with ensuring that you have the relevant background and experience for the project, is a major component of Junior Seminar. There have also been students who change their project during Senior Seminar, and still finished their comps within the semester; although this is not recommended, this delayed timeline shows how much leeway there is for exploration and backpedaling.

As a rough guide, comps projects should be equivalent to the work of an upper-level CS elective, and should either dive deeply but narrowly into a subfield of CS, or reach broadly across subfields or across disciplines without being too shallow. The only constraint on the topic is that CS+Math and CS+X students must select a project that fits within their theme, and should check with their academic advisor to make sure this is the case. Although double majors could do both comps on the same topic, they must have aspects that are unique to each major. For example, a cognitive science and computer science double major could do both projects around biologically plausible neural networks, but the cognitive science project might be about comparing its activity to real brains, while the CS project might instead focus on its algorithmic performance. We will spend more time on this in class, but you might consider the following prompts in brainstorming your project:

- What topics were you most interested in from your classes? What area of CS would you like to dive deeper into?
- Alternately, what topic were you unable to take a class in (either due to full enrollments, or because Oxy doesn't offer it)?
- What interdisciplinary topics or CS-adjacent topics would you like to explore?
- What projects might be a good showcase of your abilities to potential employers in your desired field?

A list of past comps projects can be found online.[1]

3 Methods

3.1 Sending Data From Arduino to Webpage

The goal of the first milestone is to successfully get Arduino to communicate with the frontend and then also program the Arduino to control a servo motor based on the amount of light that it receives. My first step is finding a way to get the Arduino to communicate with the frontend. After researching the different ways to accomplish this, I found that running the entire process through a Nodejs server would give me the best chance at success. The process for how the Arduino is able to communicate with the frontend through the Nodejs server is outlined in Figure 1. The Arduino sends and receives data to and from the server through the serialport. The frontend sends and receives data to and from the server through websockets. In this first case my plan is to write a program for the Arduino where I will activate a switch on the Arduino and when this switch is activated, it should change the color of a square on the webpage.

The other goal is

Beyond the basic syntax, much of learning LaTeX is learning the different commands and environments that exist. For example, text can be made **bold** with `\textbf`, *italic* with `\textit`, and `monospaced` with `\texttt`

[1] `https://www.oxy.edu/academics/areas-study/computer-science/past-senior-comps-projects`

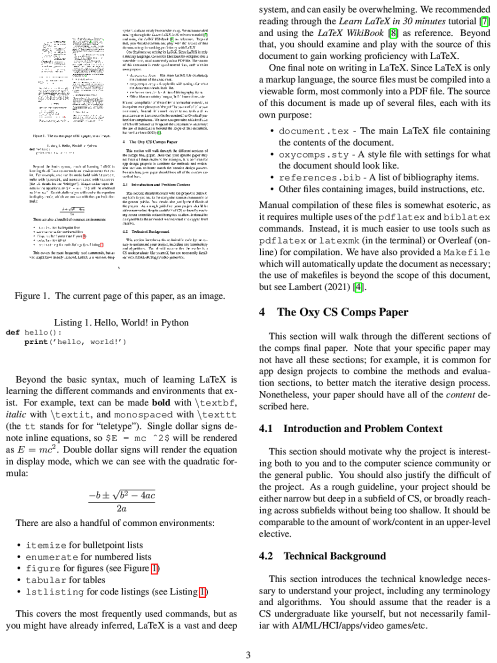

Figure 1: The current page of this paper, as an image.

Listing 1: Hello, World! in Python

```
def hello():
    print('hello, world!')
```

(the `tt` stands for for "teletype"). Single dollar signs denote inline equations, so `$E = mc ^2$` will be rendered as $E = mc^2$. Double dollar signs will render the equation in display mode, which we can see with the quadratic formula:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

There are also a handful of common environments:

- `itemize` for bulletpoint lists
- `enumerate` for numbered lists
- `figure` for figures (see Figure 1)
- `table` and `tabular` for tables (see Table 1)
- `lstlisting` for code listings (see Listing 1)

This covers the most frequently used commands, but as you might have already inferred, LaTeX is a vast and deep system, and can easily be overwhelming. We recommended reading through the *Learn LaTeX in 30 minutes* tutorial [7] and using the *LaTeX WikiBook* [8] as reference. Beyond that, you should examine and play with the source of this document to gain working proficiency with LaTeX.

One final note on writing in LaTeX. Since LaTeX is only a markup language, the source files must be compiled into a viewable form, most commonly into a PDF file. The source of this document is made up of several files, each with its own purpose:

- `document.tex` - The main LaTeX file containing the contents of the document.
- `oxycomps.sty` - A style file with settings for what the document should look like.
- `references.bib` - A list of bibliography items.
- Other files containing images, build instructions, etc.

Manual compilation of these files is somewhat esoteric, as it requires multiple uses of the `pdflatex` and `biblatex` commands. Instead, it is much easier to use tools such as `pdflatex` or `latexmk` (in the terminal) or Overleaf (online) for compilation. We have also provided a `Makefile` which will automatically update the document as necessary; the use of makefiles is beyond the scope of this document, but see Lambert (2021) [4].

4 The Oxy CS Comps Paper

This section will walk through the different sections of the comps final paper. Note that your specific paper may not have all these sections; for example, it is common for app design projects to combine the methods and evaluation sections, to better match the iterative design process. Nonetheless, your paper should have all of the *content* described here.

4.1 Introduction and Problem Context

This section should motivate why the project is interesting both to you and to the computer science community or the general public. You should also justify the difficult of the project. As a rough guideline, your project should be either narrow but deep in a subfield of CS, or broadly reaching across subfields without being too shallow. It should be comparable to the amount of work/content in an upper-level elective.

4.2 Technical Background

This section introduces the technical knowledge necessary to understand your project, including any terminology and algorithms. You should assume that the reader is a CS undergraduate like yourself, but not necessarily familiar with AI/ML/HCI/apps/video games/etc.

4.3 Prior Work

This section describes of related and/or existing work. This could be scientific or scholarly, but may also be a survey of existing products/games. The goal of this section is to put your project in the context of what has already been done.

4.4 Methods

This section describes what exactly you will be working on. What are you building? How will it combine/incorporate ideas from the literature? Be specific about what you will be doing: talk about the specific algorithm you will implement/use, the specific dataset/platform/API, and what the outcome of your project will look like. All of these decisions should be justified as well.

4.5 Evaluation Metrics

This section describes how you will evaluate your project. What will you be measuring, and how will you measure it? You might think about what would result in an F, a C, or an A for comps. Alternately, think about what are the minimal requirements for passing the class, what you might do if you had more time and resources, and what the best case scenario would be if everything went swimmingly.

4.6 Evaluation Results and Discussion

limitations

4.7 Ethical Considerations

Are there any ethical concerns that might arise from your project? You might think about whether your project perpetuates societal inequity (or could be used by others to do so), whether the data/platforms you are using is collected with informed consent and free of bias, and whether you might be subject to technological solutionism instead of working support/better the public infrastructure. Include a discussion of how you plan to mitigate these issues in your project.

4.8 Future Work, and Conclusion

4.9 Timeline

A timeline of major milestones, with specific items to be completed by specific dates/months. Note that this timeline must start over the summer; otherwise you will unlikely have enough time to complete a project of the expected scope. As part of the discussion around your timeline, talk about what you already know that would help you with the project, and what you expect to have to learn to be successful. Include programming languages, technical concepts, as well as processes (e.g., user testing).

4.10 Code Documentation

This section will demonstrate that you have thought through the basics of how your code will work. You should include a diagram of the overall data flow of your program, including what the inputs and outputs of each component will be, and how they will be represented.

4.11 Appendices

5 Tips and Advice

References

[1] Alton, Larry. *Why Every Developer Should Know a Bit of Technical Writing*. 2020. URL: https://www.computer.org/publications/tech-news/trends/why-every-developer-should-know-a-bit-of-technical-writing.

[2] Google. *Technical Writing*. URL: https://developers.google.com/tech-writing.

[3] Indeed. *11 Important Soft Skills for Software Developers*. 2021. URL: https://www.indeed.com/career-advice/career-development/software-developer-soft-skills.

[4] Lambert, Chase. *Makefile Tutorial*. 2021. URL: https://makefiletutorial.com/.

[5] Mei, Derek. *Why developers should know how to write*. 2018. URL: https://www.freecodecamp.org/news/why-developers-should-know-how-to-write-dc35aa9b71ab/.

[6] Occidental College. *Comprehensive Requirement*. 2022. URL: https://oxy.smartcatalogiq.com/2022-2023/Catalog/Academic-Information-and-Policies/Bachelor-of-Arts-Degree/Comprehensive-Requirement.

[7] Overleaf. *Learn LaTeX in 30 minutes*. 2021. URL: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes.

[8] WikiBook. *LaTeX WikiBook*. 2022. URL: https://en.wikibooks.org/wiki/LaTeX.