Final Presentations and Projects

Starring in your own movie

We have 40 projects to be presented in the last two class sessions. To do this efficiently and effectively, we're going to do what other oversubscribed courses are doing, and ask you to make a movie. The videos will be *no more than 3 minutes long*. (The reason we're sure about this is that we will show only the first 3 minutes no matter how long it is.) You will then have a few moments to bask in the wild applause from your classmates while we rewind your video and start the next one.

You should upload your video to Youtube by *midnight the night before your presentation*, then annotate the usual Google spreadsheet with a link to it. That way we can move from presentation to presentation smoothly. The sheet will be here (when we're ready this will be a live link to the sheet). All videos will be played from my laptop, so they should not be private on Youtube.

If for some reason you can't make a video (smartphone problem, etc.), remember that you can easily add a narration soundtrack to a PowerPoint presentation, then turn that into either a PowerPoint show (i.e., a self-running presentation) or a video (mp4 or wmv; make sure it runs on a pc). By whatever means, you need to create a 3 minute presentation with its own narration soundtrack that runs by itself.

Presentations will be on the 14th or the 16th. We know this is a busy time of year and will work with you to create a schedule that tries to take into account any *unique* constraints you may have. (Note: *I have a lot of work to do* is not a unique constraint.)

You will also need to write a term paper. The ability to communicate your ideas is an important skill, and you need to be able to do this at varying lengths and in varying levels of detail. Think of the movie as the trailer for the term paper – it should convey the most important ideas and results. The paper should say these things as well, but go into considerably more detail. Being able to describe your projects in a coherent write-up is an important skill, hence the multi-modal (hah!) requirement of both presentation and written report.

For all things you turn in regarding the final project, *one submission per team* is all you need do. No need for each team member to upload the same thing. Just make sure all team member names are on each upload.

Outline of the movie

The movie should take advantage of the fact that people have heard about your project before, so you need only remind them of the basics, briefly and quickly.

- What is the system supposed to do? (~10 seconds)
- Focus on an impressive demo with a *real example* of input and the system's response.
- Describe how it works. Note that this should be at the conceptual and architectural level, not at the level of code. Describe what the major ideas, techniques, and tools you used.
- Describe how well it works. What did it do well and what did it do badly? Provide an example of *an interesting failure*. recognizes everything as a circle

Outline of the Paper

There is obvious overlap here with movie, in the same sense that the full length movie repeats the trailer, but has a lot more. The paper is expected to be much more detailed and technically informative. Note that your write-up ***must*** all of the sections listed here. Really, all of them. You may have other sections as well, but you have to have each of these with substantial material in them.

The emphatic tone here arises from the fact that some people decided to ignore pieces of the project proposal outline. Don't do that here. Have something substantive to say for each section. That's a requirement.

Generalities/Logistics

1) Collaboration on the project should be detailed in the write-up: indicate who had the major responsibilities for which parts of the project in building and testing the system. If everyone was involved in most everything, that's fine; say so.
2) Papers (in pdf) are due 7PM on Friday May 17, uploaded to Stellar in the usual way.
3) Papers should ***also*** be turned in by *hardcopy* by the same deadline, in Prof. Davis's office (32-237). We ask you to print it because then each of you does 10 minute's worth of work, rather than the course staff doing a few hours' worth of work. This also avoids the inevitable difficulty of getting fonts and/or figures to print properly. Make sure all members of the team are listed on the title page.
4) Expected length is 10-15 double spaced pages, covering the material listed. *Papers will be judged on content, insight, etc., not length*. If you can be brief and substantive, fine; just make sure you address the issues above. Conversely, 20 rambling pages will not be well received

Required Sections. There is a lot of detail here, intended to help guide you.

1) Cover page
   The cover of your term paper should be a graphic and an abstract describing the task, what you built, what hardware/software it used, how it worked, and how well. See below for an example. The graphic can be a frame from your movie, a screen grab from a demo, a photo of the user using the system, etc. Anything that gives a good feeling for what you created is fine. Include all team member names.
2) Introduction and Overview
   a) What is the task? Motivate it – why is it interesting?
3) Describe the system
   a) What does it do? Show a real example of input and the system's response.
4) Describe how it works.
   a) Note that this should be at the conceptual and architectural level, not at the level of code.
   b) Describe how well it works. Do the best you can to *explain why* it had this performance. What experiments did you do? What did it do well and what did it do badly? Provide an example of *an interesting failure* and explain in detail what the failure was, what made it interesting, and what you learned from it.
   c) What in the implementation turned out to be more difficult than expected and why? (e.g., the Leap sensor was insufficiently accurate in some way, which you should describe). What did you do about that difficulty?
   d) What worked easily?

5) If you had to modify the project as you went along, where and why did you have to pivot? Explain any roadblocks you ran into and what you tried to do to get around them, before deciding that you had to modify the project.
6) You've gotten feedback from in-class demos. What if anything did you change in your system in response to the feedback?
7) User study.
   In previous years we have required a pilot user study where you had to test the system on five volunteers and write up the results. This term was a little rushed so we have eliminated that as a requirement. If you have already done a user study of any sort, excellent, report the results here. Otherwise please *design* the user study you *would* do to test some aspect of your system and include that here. The second part of this document details what a user study design should contain.
8) Performance
   a) A crucial thing about the project is *what you learned from it*, whether or not it worked perfectly. Good performance that you can't explain isn't worth much, while disappointing performance that arises for an interesting reason can be quite important.
   b) Give an example that's just out of reach for your system – what's an interesting next step that it can't quite do. As mentioned in class, this is an excellent way to explain both the power and the limits of what you've created, and set the stage for the next person's work.
   c) Describe what you would do next to improve the system, to take its performance to the next level.
9) List each of the tools/packages/libraries you used. *This is very important* and helps to guide the students taking the course next term, as their feedback was used to guide you.
   a) Provide current url's for access to each of these tools.
   b) Explain what they did and how well they worked (it's useful to
   c) Indicate whether you used them just as they were, or had to modify them. If so, indicate in general terms what you had to do.
10) Implementation review
   a) What in the implementation turned out to be more difficult than expected and why? (e.g., the Leap sensor was insufficiently accurate in some way, which you should describe). What worked easily?
   b) If you had to modify the project as you went along, where and why did you have to pivot?
   c) Describe what you would have to do to improve the system to take its performance to the next level. Include an assessment of how hard this would be by placing it somewhere on the spectrum from routine software engineering to creating a fully sentient AI.

Turning in Code

1) A github is a fine way to submit the code for your project. If it's small, you can also submit a zip file to Stellar. In either case be *sure* you have included everything needed.
2) The code should be *runnable* by us if at all possible, i.e., unless it requires specialized hardware other than a Leap or Kinect. In that unlikely case, please consult with us about what to turn in. We'll work with you to get something reasonable that we can still try out, if possible.
3) ***Be sure*** to include a README file that provides:

a) a list and explanation of each file i.e., basically a table of contents, so we don't have to figure out what's what.
b) Instructions for setting up and running your code, making careful note of what's needed to get it running on a bare machine (e.g., one with just the appropriate OS). This too is a useful skill – it's quite frustrating to get a program that's advertised as requiring a certain foundation and then discover a few hours into trying to make it work that the author forgot to mention something crucial (e.g., it runs in Python 2.7, not 3.5). So think carefully about the infrastructure your application needs, and list all of it.

Grading

A mix of several things go into the grading of the term project, in no particular order:

- creativity of the basic concept
- the soundness of the concept and design, i.e., does what you are trying to do make sense, did you assemble the right collection of tools, etc.
- does it stand out, i.e., it is something noticeably beyond an application of one of the mini-projects?
- how well does it work?
- what did you learn from it? We're less concerned about whether it worked perfectly. Good performance that you can't explain isn't worth much, while disappointing performance that arises for an interesting reason can be quite important.
- How did you handle barriers that you encountered, i.e., did you recognize a need to pivot and how well and how appropriately did you do it?

Hardware

You must turn in any borrowed hardware in person to any of the TAs or the course secretary. You are responsible until we get it back. "I gave it to a friend to turn in" will not get you off the hook. Replacing these is time consuming and expensive, so we will unabashedly hold your grade hostage until we get your hardware back.

Sample Cover Page

Fencing Coach
Janet Epee, Joseph Foil

We built a fencing coach by combining the Kinect's ability to track body pose and motion, with our knowledge of the sport. The system checks (and possibly corrects) the user's stance, then names aloud a fencing move (lunge, parry, riposte, counter-attack, etc.), observes the user's actions, and comments on them ("well done", "your right elbow was too low", etc.). It knows about 8 moves and can make 14 different critiques.

The system's accuracy rate in stance detection was …, its recommendations for improvement were correct X% of the time, …

It worked quite well for stance detection and correction, but was somewhat weaker in critiquing moves. This was due in part to the difficulty a classifying a motion as one of the moves, and the difficulty in detecting the details of a movement. ….

User Study Design

Design
Based on what you learned in class and what you read in the Nielsen Norman Group handout, design a user study for your term project. You should focus carefully on one or two selected aspects of your system. This is a good exercise: the ability to pick the right question(s) to focus on in a user study is an important skill to practice.

Another key point is that user studies are experiments, and experiments provide convincing evidence to the extent they are carefully controlled. Keep that in mind as you design the study and recall some of the sources of variation mentioned in the lecture.

While some user studies are concerned only with qualitative data (e.g., people's opinions), it's often quite useful to make a number of measurements of their behavior. Assuming you've determined what question you want to answer (as in the first above paragraph), you then have to decide what to measure. In doing this keep in mind that there's no need to invent obscure metrics; some important things to measure can be quite simple and commonsensical. The crucial idea is to be able to come away with something more than just a gut-level impression of user's performance.

Given the range of projects in the class, there is no one template for what to do. Generally speaking though we can imagine three general classes of study designs:

A) Does the design of this system make sense to people?
B) How well can they carry out a specified task?
C) Is design variant A better than design variant B?

Does the design make sense?
Consider a system designed to allow people to compose music with gestures. You might give someone a brief overview of the system's capability and operation, then determine whether they understand what the system does and how to use it. Part of the task here is to specify how you will determine whether they understand what it does and understand how to use it. What will you measure, what will you report?

How well can they carry out a specified task?
Give them an intro to the system, and perhaps a warmup exercise, then a list of three (4, 5, 6…) specific tasks to carry out. What will you measure to determine how well they do these tasks?

Is design variant A better than design variant B?
If you've been considering a couple of design alternatives, you can run an A/B study, in which you give two different groups of subjects different versions of the system. Again a key is determining what to measure – what effect do you think the design variant will have? Speed? Memorability? User appeal?

Some generic issues to keep in mind:

How will you describe your system to the subjects? How will you tell them what it does and how to use it? How will you make sure each person's introduction is the same? Once they are using it, will you answer their questions? If they ask you to, will you show them how to do something once they've started using it?

Planning the Study

Good planning is frequently the key to doing a good job on something; with a user study it's absolutely crucial. Far too often people get a fair way through their study only to realize they are not asking the right questions or making the right measurements. So plan this out carefully; it will be well worth your time. Use this as a guideline for some of the things (but not necessarily everything) to think through before you begin.

1) Characterize the kind of study you're doing, based on the framework in the Norman/Neilsen article, locating it on all three of their dimensions.

2) Specify the question you are trying to answer. For many projects the question will be some variation of how usable the system is, but be more specific. Are you trying to find out about
    a. naturalness (how much you do or do not have to explain)
    b. learnability (once they understand, how much effort to get proficient)
    c. efficiency
    d. safety
    e. accuracy (how often does it recognize what the user is trying to convey)
    f. etc.

3) What precisely are you going to ask them to do?

4) Specify what measurements/observations you will make.

5) Write the script you are going to use, including precisely what you're doing to say to the subjects. This is important – giving each subject the identical instructions is one crucial part of carrying out a controlled study. It's enormously tempting to revise the intro as you go along, as you find out what does and doesn't explain things well. If you do that you're polluting your data.

It's almost always useful to end the study with Likert-style questions around the issues you are trying to examine, e.g.,

The set of gestures seems simple and natural
It was difficult to do what I wanted to do
…

Note that good practice with Likert questions involves phrasing questions so that the "good" end of the scale appears randomly on the left and right ends. This will make people think more carefully about their answers.

Similarly, it's also useful to end the study with open-ended questions:

What did you like most
about the system? What did
you like least?
What else ought it to
be able to do? … etc.

that not every question is relevant to every project, but make a good faith effort).