

Project Showcase

Attendance!

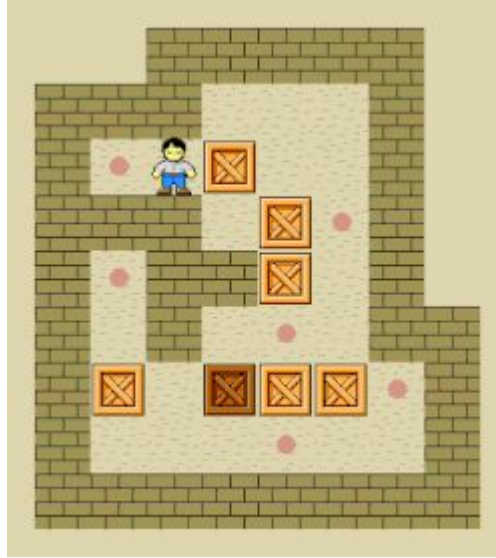
<http://bit.ly/4uCrV5i>

Homework 7

- Experience what it's like to pick up, read, and continue someone else's code, and learn why clear documentation and thoughtful organization matter so much
- In the real world, you will almost never write code entirely from scratch in isolation. You'll join teams with existing codebases, inherit projects from colleagues who've moved on, and hand off your own work for others to maintain.
- The ability to write code that someone else can understand, and to productively navigate code written by someone else, is one of the most important (and most underrated) skills in software development.
- AI tools can help with both of these challenges, but only if you use them strategically

Project options

1. Fireboy and Watergirl
2. Pac-man
3. Sokoban
4. Tower Defense



Guidelines - Phase 1

- Start coding! Use AI tools to help you scaffold the project, generate boilerplate, and write game logic. But remember: someone else will need to read this code. Prioritize clarity over cleverness.
- Create documentation as you go. This is critical. Add comments explaining your design decisions, what each major section does, and what still needs to be done. Reserve enough time in class to create a README that describes the project, how to run it, what's been completed, and what remains. (See the tips section for specifics.)
- After ~30 minutes, stop working. Commit and push your current code to GitHub, even if it is incomplete. This is expected! The goal is a strong, clear starting point, not a finished game.
- You will not continue working on your own Phase 1 code after class. Your Phase 1 code is your gift to another student. Make it a good one.

Guidelines - Phase 2

- Clone or download the repository you receive.
- Before changing anything, make sure to commit the initial state of the code you received to your own repository. This is important so that we (and you) can see exactly what starting point you were given. (You can fork the repo, or create a new repo and make an initial commit with their unmodified code. The latter is probably easiest.)
- Read the code and README carefully. Try to understand the structure, what has been done, and what remains. Use AI tools to help you understand unfamiliar code.
- Start making progress. Fix bugs, add missing features, complete unfinished functions, or polish the app if it is already working. Work strategically: identify what will have the most impact and focus there.

Guidelines - Phase 2

- You must retain at least 75% of the original code. You may reorganize, refactor, add to, and fix the code, but you cannot simply delete or comment out most of it and start over. The point is to work with the existing code, not to replace it. (We will compare your final submission against the initial commit to verify this.)
- Stop soon after one hour. It's okay if the project is not complete! Commit and push your final code.
- Update the README to describe the following: what you changed, what challenges you faced, and what you would do next if you had more time.

Reflection

Reflection prompt: As you work through Phase 2, pay attention to what makes the code easy or hard to work with. Was it well-commented? Were variable names clear? Was the structure logical? Did the README help? These observations are valuable and will shape how you write code going forward, both for yourself and for others.