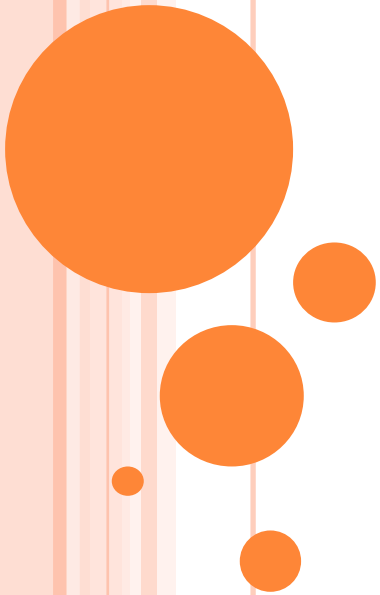


EECS22L Lab Discussion Week 2

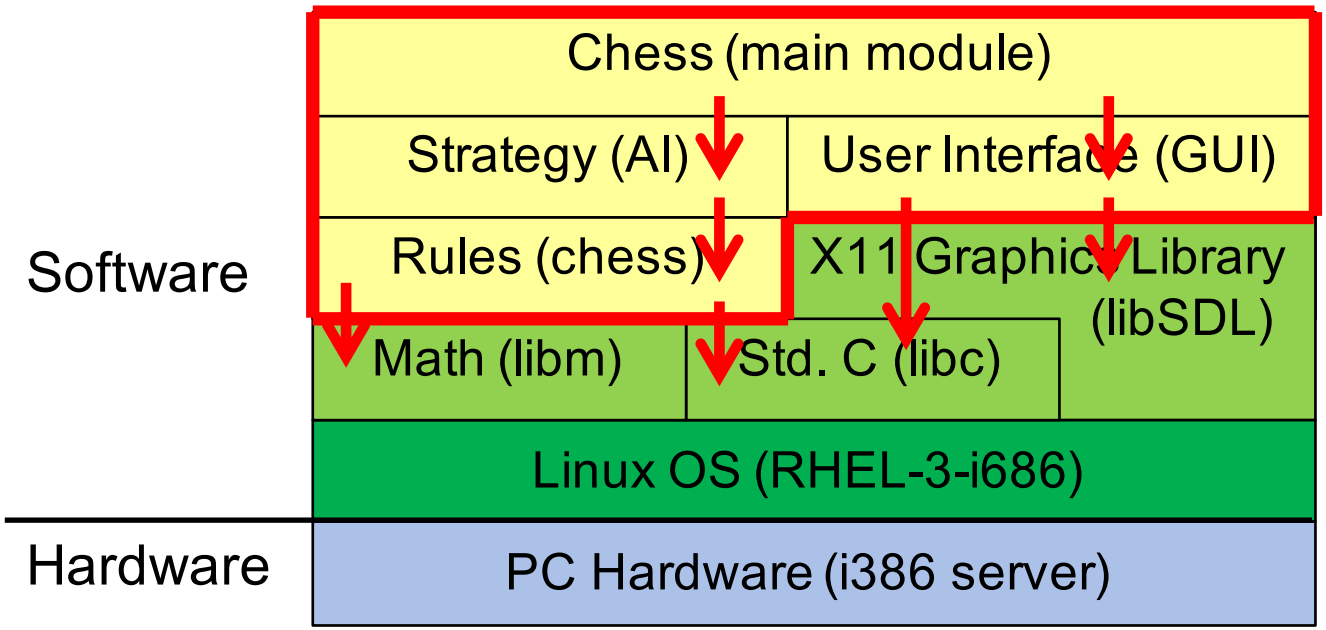
Grading Notes

Some problems of your first submission

- Software features section does not have to include program level APIs as DrawBoard(), MovePiece(), these should be in software spec
- Software installation, should be a Linux-based program, instead of a Windows-based one, one team asks the user to double click "install.exe" and "uninstall.exe"
- One team has an unreferenced image (https://www.codeproject.com/Articles/36112/Chess-Program-in-C)
- Software uninstallation: "Delete the directory", which directory to be deleted is not specified; no need to restart computer
- Setup and config: too detailed for different platforms: Windows, mac, Linux, as it is a Linux-based program; no need to explain how to connect to a server via Windows/Mac in great length
- One team's user manual has a nice GUI, but installation/uninstallation is not well-explained; missing copyright and error message part
- Improper short names: human vs CPU (may refer to Central Processing Unit), using AI/computer instead is better
- Need to specify whether to use keyboard or mouse to play the game

Project – Chess Game

- Example: Software Layers and Modules
  - Stack of all components in the software architecture
    - Hardware infrastructure
    - Operating system (OS) infrastructure
    - OS and third-party libraries
    - Application modules



Project – Chess Game (cond.)

- Required Features in Chess Game
  - Following the official rules of Chess
  - Game Interface (load/save game, game mode selection)
  - Board Display (with ASCII or GUI)
  - Keep the log of all moves in the game
  - Board Setup (in normal or specific way)
  - Different levels of Auto-player (easy to hard)
  - Smart Auto-player (smart AI)
- Desirable Features in Chess Game
  - Timer
  - Hints
  - Audio

Project – Chess Game (cond.)

- Design the software architecture specification
  - Modules (in header files)
    - How many modules will you need?
    - What are the APIs for each modules ?
  - Data structures (in header files)
    - How to represent the chess board?
    - How to represent the chess pieces?
    - How to represent moves?
  - Algorithms (on paper first, maybe?)
    - How to keep track of the moves?
    - How to make a random move?
    - How to make the smartest move within one step?
    - How to make the smartest move within two steps?
    - …

Project – Chess Game (cond.)

- Let's use hw5 (Movielab) in EECS22 as an example
  - Modules (in header files)
    - How many modules will you need?
      - module which reads the video into frames
      - module which creates video stream out of those frames
      - module which manipulates the frames to create v-flip, h-flip, or black & white video
      - ……
    - What are the APIs for each modules ?
      - CreateMoive:
        Input: number of frames, height and width of image
        Output: pointer to the data structure storing video
        Description: allocate memory for the movie and the memory space for the frame lists
      - YUV2RGBImage:
        Input: pointer to YUV image and RGB image
        Output:
        Description: convert pixels in YUV to value in RGB

Project – Chess Game (cond.)

- Data structures (in header files)
  - How to represent the video stream?
    - Double-Linked List
  - How to represent the image in the video stream?
    - three 1-d arrays for R/Y, G/U, B/V pixels.
- Algorithms
  - How to convert image to image in Black&White?
  - How to create video playing in reversed order?
  - How to generate Mandelbrot images?

Project – Chess Game (cond.)