Mouse Emulation

Dry-Running Mouse Code and World Simulation

Rob Probin
Minos, April 2017

Concept

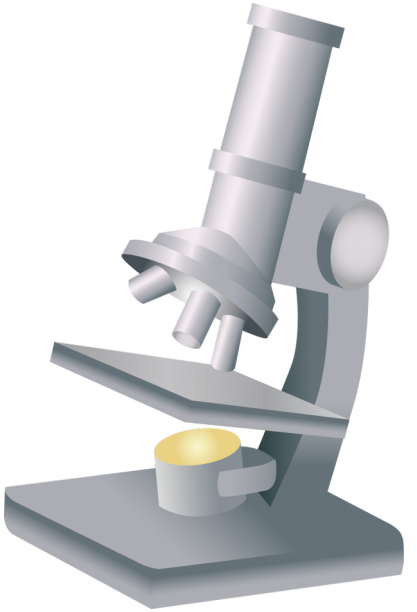
- Test as much as practical on a PC
- Allow testing on both PC and on the mouse
  - With the same mouse code

Why?

Mouse development can be slow!

- Time to download new software
- Getting the mouse into a position to test
- Test runs
- Batteries finite
- Tools & Visualisation poor
- Difficult to get exact reproducibility
- Eliminating basic logic problems can be difficult

BUT *Still need testing in a real maze*



What **is** practical?

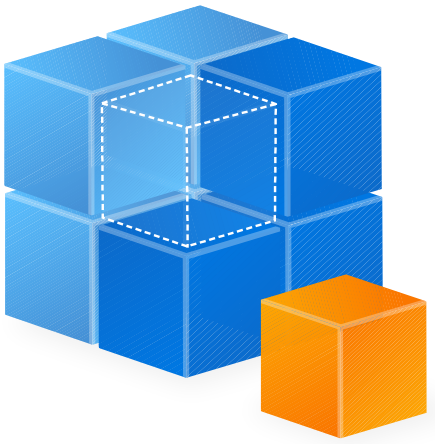
- Time creating simulators vs. time on mouse
  - Few MCU / MPU have accurate on-chip peripheral simulators
  - Mouse specific hardware – write simulator?
  - Real world physics - tricky
  - Mouse specific parameters – unknown
    - Without a lot of work
    - Masters-level project?

What do we want to?

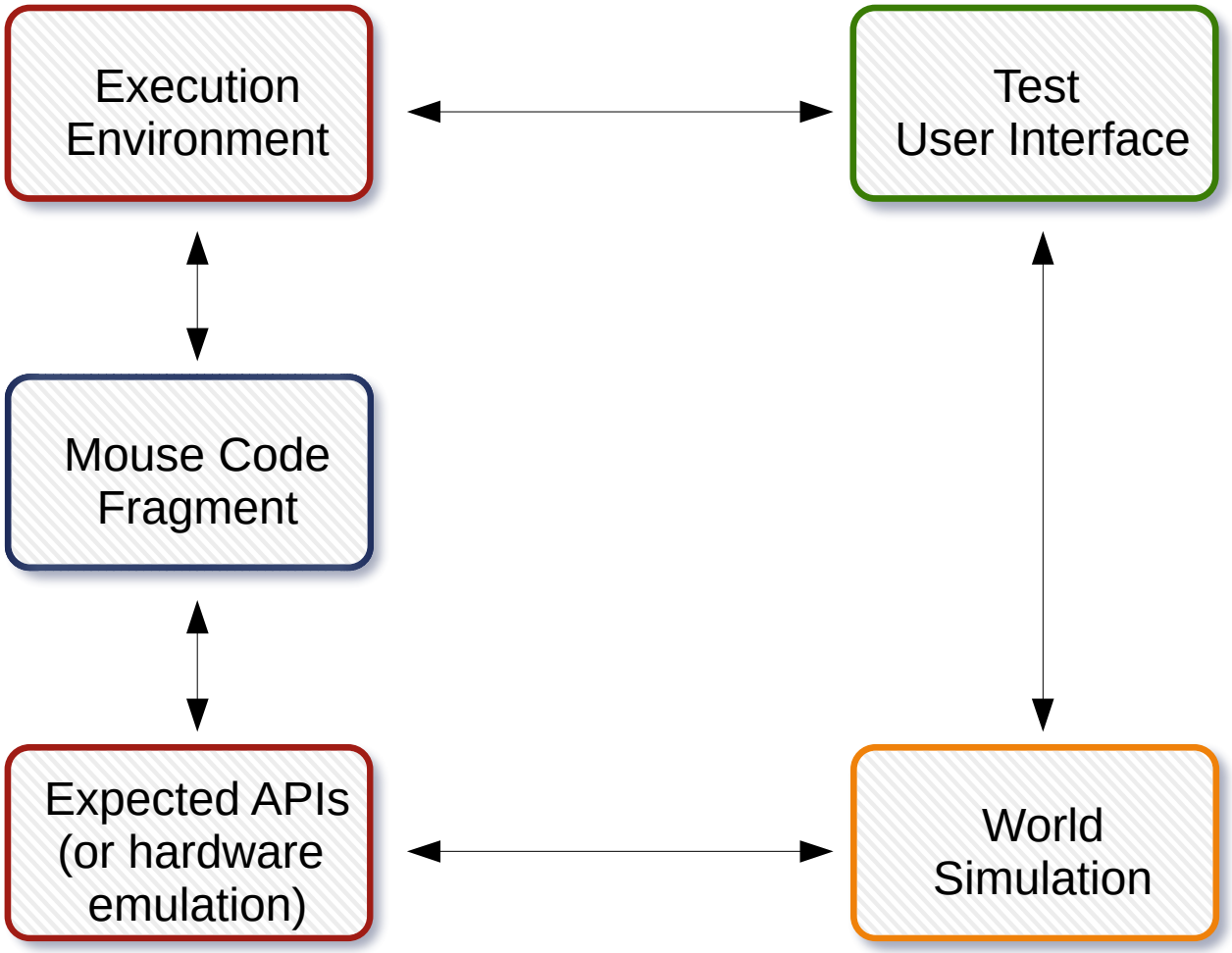
- Emulate a run-time environment for the mouse code to run
  - successfully in a simulation of the world
- Simulate the world accurately enough to allow the mouse code (fragment) to function
  - Be tested within confined limited parameters
  - With sufficient detail based on the purpose of the simulation

Areas to Consider Testing

- Maze solver algorithm testing
- High level control
- Low level control (I/O, etc.)
- Physics
  - Basic stepped motion / discrete position
  - Real position
  - Acceleration, etc.
  - Characterising a mouse vs. estimated simulation?
- Auxiliary
  - e.g. battery usage based on activity, etc.



Generic Architecture



Other things to consider

- Text display?
- Graphic display?
- Assembler mouse
  - Emulator?
  - vs. high-level mice: C, C++, BASIC, Python, etc.
    - (Cross compile or native running)
- Time…

Real-time?

- Good, but not essential?
  - Generally simulators don't need to be real-time
  - Emulators should be real-time?
  - Obviously time inside the simulator should be consistent, at least for some purposes!
  - Not sure … implementation … computer time vs. synthetic time
  - Computer time might give us reproducibility problems (if we are not careful)
    - Some network games suffer with a similar problem when running multiple sims
- Rewind time? Slow time? Forward/Back?
  - Reversible functions?
  - Repeatable function?
  - Accurate step-by-step logging

Retro Fitting or Design for test?

- Retro Fitting:
  - Most likely case for us
  - Can alter mouse code slightly?
  - Is separation of concerns possible
    - to test less than everything?
  - Not ideal – might want to refactor code
- Designed for test:
  - Need to think about this!
  - Proper abstractions between layers
  - Low coupling
  - Allow test stubs to be inserted (by pointer or by compilation)

My Simulator – Mk 1

- Flood-fill maze solver testing
- Visual inspection of results, mostly
- Some automatic – e.g. failed to solve
- Various 16x16 and 5x5 mazes

My Simulator – Mk 2

- Text output
  - from mouse and from simulator
- Keyboard input
- Discrete cell step
- Basic high level code
- No I/O
  - Emulated serial link between the two MCUs
- No need for real-time (mostly)
- Very slight code change
- Supports different mazes

My Simulator – Mk 3 (in progress)

- Graphical output
  - LEDs, Maze & Mouse (in progress)
- GUI buttons
- Floating-point cell step (in progress)
- Basic high level code and some low level code
- No I/O yet
  - Emulated serial link between the two MCUs
- Based on text simulator
- Some code changes to pass GUI instance to Text

Questions? Comments?