Mastering Rodents

2-bit programming of 8-bit mice

Bored Already?

https://github.com/blondie7575/MouseII

Searching for modernity

- Firmware, firmware, where to put the firmware?
- Screen holes! Some readonly!
- Slot firmware areas!
- All of them!

Searching for modernity

- Old school ROM entry points
  - JSR $fded
- Modern style API calls
  - Indirection in the name of abstraction

Spanning generations

- Apple //e Enhanced and Apple //c quite different
- Firmware moved in nearly every //c version
- Many other pain points here, so stay tuned
- Lots of self-modifying code

Search for pellets

- Apple //e
  - Search for magic pattern $38 18 01 20 d6
    - Yes, seriously
- Apple //c
  - The same!

Squeak!

- ROM entry: $cX00 + routine index
- Indirect jump
- X = routine number
- Y = slot number
- Not re-entrant!!

Better mousetrap?

- Mouse is fundamentally asynchronous
- Interrupts are your friend
- On //c, they're interrupts anyway
- Independent of clock speed
- Included code used Combined Interrupt Mode

Better mousetrap?

- SERVEMOUSE
- X,Y values 0-1023
- Set clamping for power-of-two scaling math

Better mousetrap?

- For example, for 80 distance in X:
  - Clamp to 640
    - lsr $0578
      ror $0478
      lsr $0578
      ror $0478
      lsr $0578
      ror $0478

Comparing Apples

- Apple //e Enhanced with AppleMouse card:
  - Onboard 6502
  - Interrupts only on new data (movement, button)
  - 60Hz sample rate
  - Button interrupts only on down event

Comparing Apples

- Apple //c and //c+:
  - CPU driven
  - VBL interrupts
  - 30Hz sample rate
  - Button interrupts always while down

Known issues

- No button up interrupt
- New button down not detected until mouse moves
- Either play nice with user on //e only, or be consistent
- Not IIgs compatible, although mostly harmless

Demo!

Any Questions?

https://github.com/blondie7575/MouseII