

# CPSC 436D

## Video Game Programming



### Basic Team Programming



© Alla Sheffer

## Gameplay



- State Machine
  - Fixed set of states, parameter based transition
  - Can encode best next move
- Behavior Tree
  - Prioritize moves based on priorities
- Strategy (search)
  - Given current state, determine **BEST** next move
  - Short term: best among immediate options
  - Long term: what brings something closest to a goal
    - Search behavior tree for exact/approximate path to best outcome

© Alla Sheffer



## Team Project Basics

- *Version Control*
- *Testing & Debugging*

© Alla Sheffer



## Version Control

*Why?*

© Alla Sheffer



## Version Control

### **Why?**

- Merge input from different coders
- Backup
- Trace bug history
- Experiment with alternatives

© Alla Sheffer



## Debugging

- ***There will be bugs...***
- ***Strategies for Fixing?***

© Alla Sheffer



## Debugging

- ***There will be bugs...***
- ***Strategies for Fixing?***
  - Anticipate
  - Reproduce
  - Localize
  - Use proper debugging tools

© Alla Sheffer



## Debugging

- ***Strategies for Fixing?***
  - Anticipate I = Test
    - *Unit tests*
    - *Logging*
    - *Explicit tests for “what can go wrong” (assert)*
      - Anything that can go wrong will go wrong... at the worst possible time
    - *For games: backdoor input/output*
    - *Visual testing (early)*
  - Reproduce
  - Localize
  - Use proper debugging tools

© Alla Sheffer



## Debugging

- **Strategies for Fixing?**
  - Anticipate II: *your compiler (-Wall) is your friend*
  - Reproduce
  - Localize
  - Use proper debugging tools

© Alla Sheffer



## Debugging

- **Strategies for Fixing?**
  - Anticipate
  - Reproduce
    - *When does it happen?*
    - *Logging + unit tests*
  - Localize
  - Use proper debugging tools

© Alla Sheffer



## Debugging

- **Strategies for Fixing?**

- Anticipate
- Reproduce
- Localize
  - *In time: version control*
  - *In place: logging*
    - Divide and Conquer
  - *Minimal trigger input*
  - *Don't guess; measure*
- Use proper debugging tools

© Alla Sheffer



## Debugging

- **Strategies for Fixing?**

- Anticipate
- Reproduce
- Localize
- Use proper debugging tools
  - *Run with debug settings on*
  - *Run within a debugger*
    - Set breakpoints
    - Examine internal state
  - *Learn debugger options*

© Alla Sheffer

## Debugging (From Waterloo ECE 155, Zarnett & Lam)



- **Strategies for Fixing?**
  - Scientific method.
    - Observe a failure.
    - Invent a hypothesis.
    - 3 Make predictions.
    - 4 Test the predictions using experiments and observations.
  - Correct? Refine the hypothesis.
  - Wrong? Try again with a new hypothesis.
  - Repeat

© Alla Sheffer

## Debugging (From Waterloo ECE 155)



### More (Human Factor) Strategies

- Take a Break/Sleep on it
- Code Review
  - Look through code
  - Walk someone through the code

© Alla Sheffer



## Debugging

### More (Human Factor) Strategies

- Question assumptions
- Minimize randomness
  - Use same seed
- Check boundary conditions
- Disrupt parallel computations

© Alla Sheffer



## Debugging (From Waterloo ECE 155)

### More Strategies

- Know your enemy: Types of bugs
  - Standard bug (reproducible)
  - Sporadic (need to chase – right input combo)
  - Heisenbug
    - Memory (not initialized or stepped on)
    - Parallel execution
    - Optimization

© Alla Sheffer





## Hard Bugs (cheat sheet)

- *Bug occurs in Release but not Debug*
  - Uninitialized data or optimization issue
- *Bug disappears when changing something innocuous*
  - Timing or memory overwrite problem
- *Intermittent problems*
  - Record as much info when it does happen
- *Unexplainable behavior*
  - Retry, Rebuild, Reboot, Reinstall
- *Internal compiler errors (not likely)*
  - Full rebuild, divide and conquer, try other machines
- *Suspect it's not your code (not likely)*
  - Check for patches, updates, or reported bugs

© Alla Sheffer