CPSC 427
Video Game Programming

Debugging

Helge Rhodin
Setup

@Helge: Pressed record?

@Class: Logged into iClicker cloud?
Logistics: Exam slot?

• *Final cross-play session*
• *Industry jury*
• *Awards*
• *Attendance mandatory*

• *Scheduled: Dec 18th, 19:00*
CPSC 427
Video Game Programming

Debugging

Helge Rhodin
Debugging

• There will be bugs…
• Strategies for Fixing?

Learning goals:
• Knowing about different debugging techniques
• When to look for what type of bug
• Strategies for avoiding bugs!
Debugging

• *There will be bugs…*

• **Strategies for Fixing?**
  • Anticipate
  • Reproduce
    • *Things get terribly difficult if randomness is involved!*
  • Localize
  • Use proper debugging tools
Task: Recall bugs that you faced

- *Those that you encountered early*
- *Those you had to track down*
Catastrophic Software Bugs…

Ariane 5 Flight 501, 4 June 1996

PacMan at level 256
Debugging: Strategies for Fixing?

- Anticipate
  - Unit tests
  - Logging
  - Explicit tests for "what can go wrong" (assert)
    - Anything that can go wrong will go wrong… at the worst possible time
  - State/play saving and loading speeds up debugging
  - Visual testing (early)
  - Avoid randomness (use seed for rnd)
- Reproduce
- Localize
- Use proper debugging tools
Debugging: Strategies for Fixing?

- Anticipate II: your compiler (with –Wall enabled) is your friend
  - “This enables all the warnings about constructions that some users consider questionable, and that are easy to avoid”
- Reproduce
- Localize
- Use proper debugging tools
Debugging

- **Strategies for Fixing?**
  - Anticipate
  - Reproduce
    - *When does it happen?*
  - Logging + unit tests
  - Record/load gameplay
  - Localize
  - Use proper debugging tools
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
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*
Exchange Experiences

- **Catastrophic failures?**
- **Debugging strategies that work for you**
  - Which ones don’t?
    - Can others make them work?

- **Elect a chair, report your groups most interesting bug and its fix**

Teams of 4
Debugging
(From Waterloo ECE 155, Zarnett & Lam)

• **Strategies for Fixing?**
  • Scientific method.
    1. Observe a failure.
    2. 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
More (Human Factor) Strategies

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

More (Human Factor) Strategies

- Question assumptions
- Minimize randomness
  - Use same seed
- Check boundary conditions
- Disrupt parallel computations
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
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
<table>
<thead>
<tr>
<th>Week</th>
<th>Day</th>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Mon</td>
<td>9-Oct</td>
<td>Thanksgiving</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>11-Oct</td>
<td>Lecture 7: Collisions and simple physics</td>
<td>Peer review A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tutorial: Face2Face grading A1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>12-Oct</td>
<td>Lecture 8: Simulation basics (makeup Monday)</td>
<td>Milestone 1</td>
</tr>
<tr>
<td>7</td>
<td>Mon</td>
<td>16-Oct</td>
<td>Lecture 9: Simulation advanced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>18-Oct</td>
<td>Lecture: Team-report M1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tutorial: Face2Face grading M1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Mon</td>
<td>Lecture: IO and observer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>23-Oct</td>
<td>Lecture: Debugging</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>25-Oct</td>
<td>Lecture: Debugging</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Mon</td>
<td>30-Oct</td>
<td>Lecture: Curves and splines</td>
<td>Milestone 2</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>1-Nov</td>
<td>Lecture: Team-report M2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tutorial: Face2Face grading M2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Mon</td>
<td>6-Nov</td>
<td>Lecture: Simple AI Cross-play M2</td>
<td>Assignment 2</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>8-Nov</td>
<td>Lecture: Two-player AI</td>
<td>Peer review A2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tutorial: Face2Face grading A2</td>
<td>(due on Thursday)</td>
</tr>
<tr>
<td>11</td>
<td>Mon</td>
<td>13-Nov</td>
<td>Break</td>
<td></td>
</tr>
</tbody>
</table>