CPSC 427
Video Game Programming

Debugging

Helge Rhodin

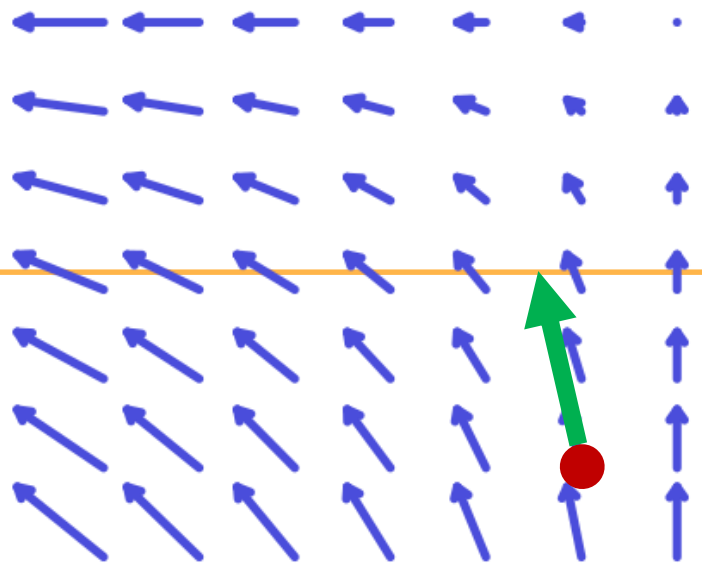
Setup

@Helge: Pressed record?

@Class: Logged into iClicker cloud?

Racap: Simulation

Force, impulse, velocity…



Our goal: position and velocity

Think of:

- ***Force as an invisible string that pulls the object***
  - *changing in magnitude and direction over time and space*
  - *without a force, the object moves in a straight line*
- ***Impulse as a change in velocity***
  (dependent on the object mass)
  - *Force applied over one timestep*
    (can be continuous or instantaneous at some point during the step)

DE Numerical Integration: Explicit (Forward) Euler

$$\frac{\partial}{\partial t}\vec{X}(t) = f(\vec{X}(t), t)$$

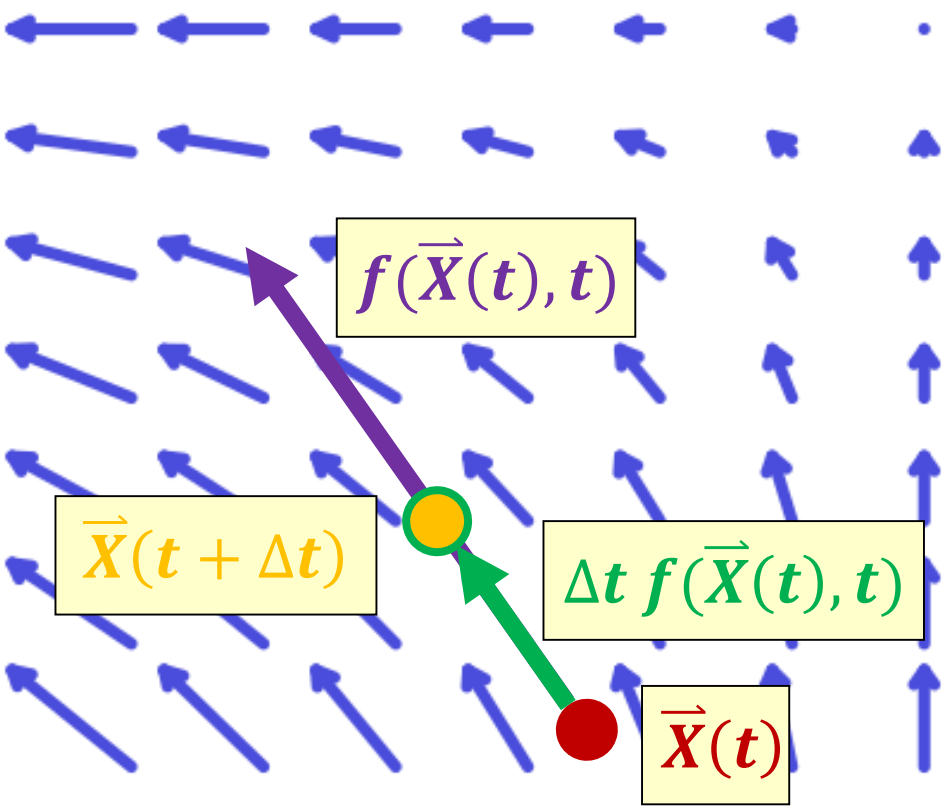
Given that $\vec{X}_0 = \vec{X}(t_0)$

Compute $\vec{X}(t)$ **for** $\mathrm{t} > \mathrm{t}_0$

$$\Delta t = \mathrm{t}_i - \mathrm{t}_{i-1}$$

$$\Delta\vec{X}(\mathrm{t}_{i-1}) = \Delta t\, f(\vec{X}(\mathrm{t}_{i-1}), \mathrm{t}_{i-1})$$

$$\vec{X}_i = \vec{X}_{i-1} + \Delta t\, f(\vec{X}_{i-1}, \mathrm{t}_{i-1})$$



Implicit (Backward) Euler:

- **Use forces at destination**

Solve system of equations

$$\frac{\partial}{\partial t}\begin{bmatrix}\vec{x}\\ \vec{v}\end{bmatrix} = \begin{bmatrix}\vec{v}\\ \Sigma\vec{F}/m\end{bmatrix}$$

$$\boxed{\begin{aligned} \boldsymbol{x_{n+1} = x_n + h\, v_{n+1}} \\ \boldsymbol{v_{n+1} = v_n + h\left(\frac{F_{n+1}}{m}\right)} \end{aligned}}$$

- **Types of forces:**
  - ***Gravity***

    $$F = \begin{bmatrix}0\\ -mg\end{bmatrix}$$

  - ***Viscous damping***

    $$F = -bv$$

  - ***Spring & dampers***

    $$F = -kx - bv$$

Implicit (Backward) Euler:

- **Use forces at destination + velocity at the destination**

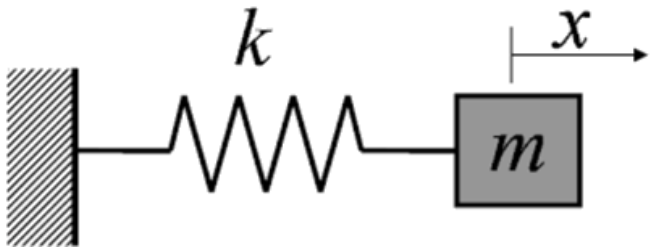
Solve system of equations

$$\frac{\partial}{\partial t}\begin{bmatrix}\vec{x}\\ \vec{v}\end{bmatrix} = \begin{bmatrix}\vec{v}\\ \Sigma\vec{F}/m\end{bmatrix}$$

$$\boxed{\begin{aligned} x_{n+1} &= x_n + h\, v_{n+1} \\ v_{n+1} &= v_n + h\left(\frac{F_{n+1}}{m}\right)\end{aligned}}$$

Example: Spring Force

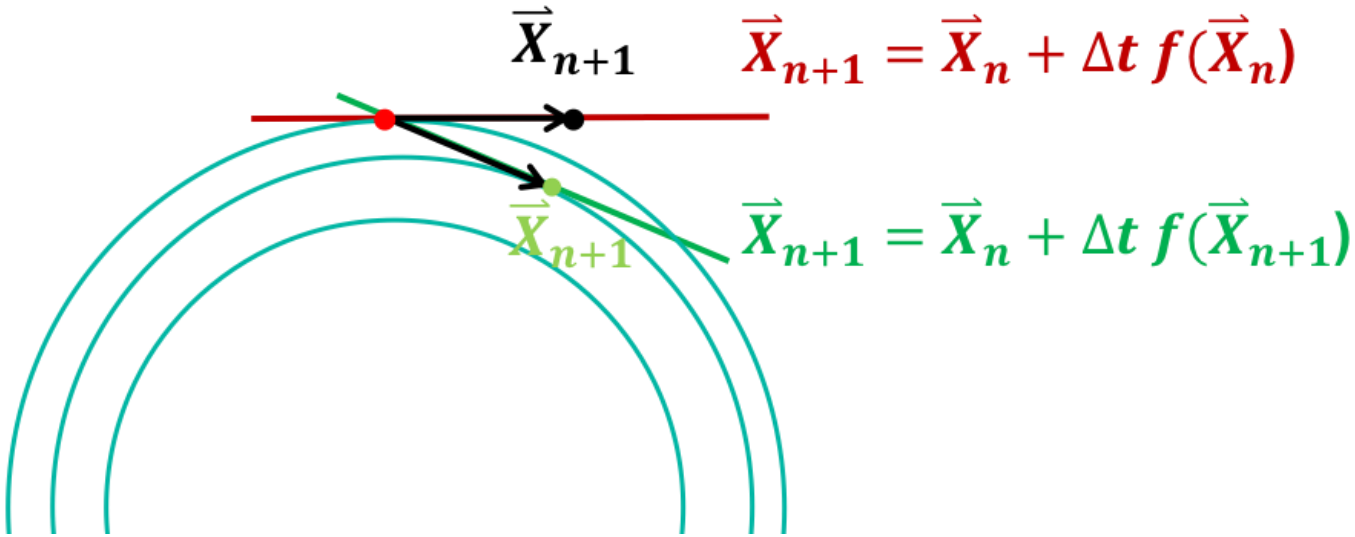
$F = -kx$

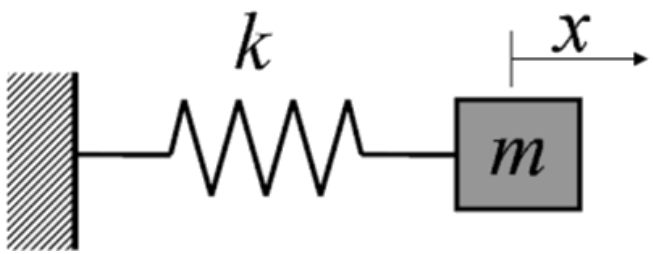


$$\boxed{\begin{aligned} x_{n+1} &= x_n + h\, v_{n+1} \\ v_{n+1} &= v_n + h\left(\frac{-k\, x_{n+1}}{m}\right)\end{aligned}}$$

Analytic or iterative solve?

Forward vs Backward





Forward Euler

$$\boxed{\begin{aligned} x_{n+1} &= x_n + h\, v_n \\ v_{n+1} &= v_n + h\left(\frac{-k\, x_n}{m}\right) \end{aligned}}$$

Backward Euler

$$\boxed{\begin{aligned} x_{n+1} &= x_n + h\, v_{n+1} \\ v_{n+1} &= v_n + h\left(\frac{-k\, x_{n+1}}{m}\right) \end{aligned}}$$

Could one apply the Trapezoid Method?

Questions

Which solver to use? For a space simulator (with accurate orbits, e.g., satellites)

1: Forward Euler

2: Backwards Euler

3: Midpoint

4: Trapezoid

5: Seq. Impulses

Questions

Which solver to use? For a jump & run

1: Forward Euler

2: Backwards Euler

3: Midpoint

4: Trapezoid

5: Seq. Impulses

Questions

Which solver to use? For a billiard game (with many balls that can stack)

1: Forward Euler

2: Backwards Euler

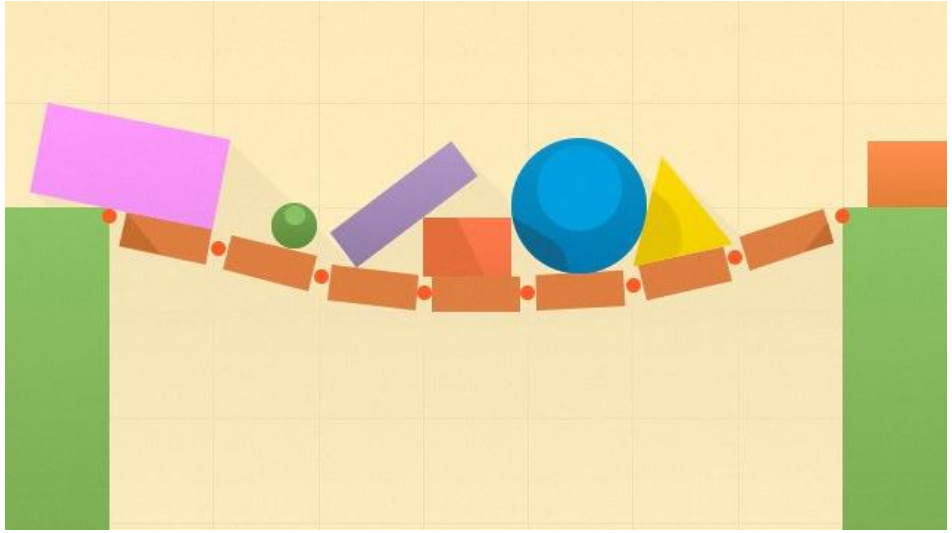
3: Midpoint

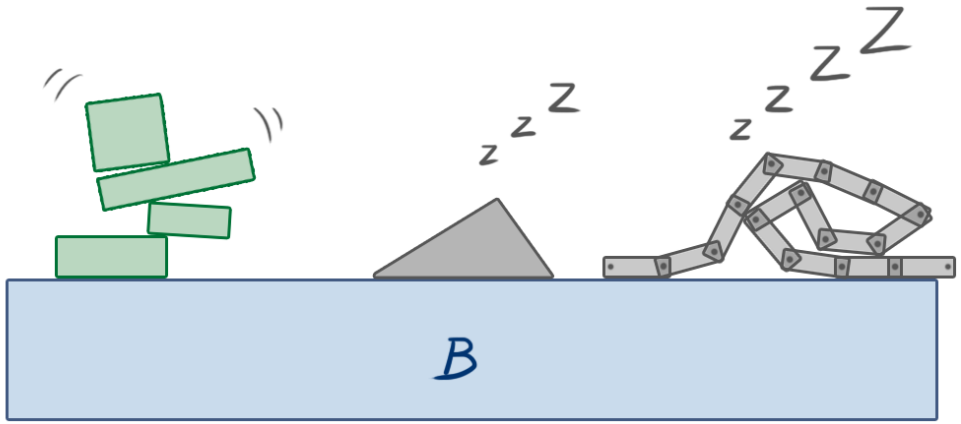
4: Trapezoid

5: Seq. Impulses

Self-study: Constrained physics

By Nilson Souto
https://www.toptal.com/game/video-game-physics-part-iii-constrained-rigid-body-simulation





Logistics: Exam slot?

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

- ***Sheduled: <u>Dec 18th, noon</u>***

- ***What else comes next?***

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…***

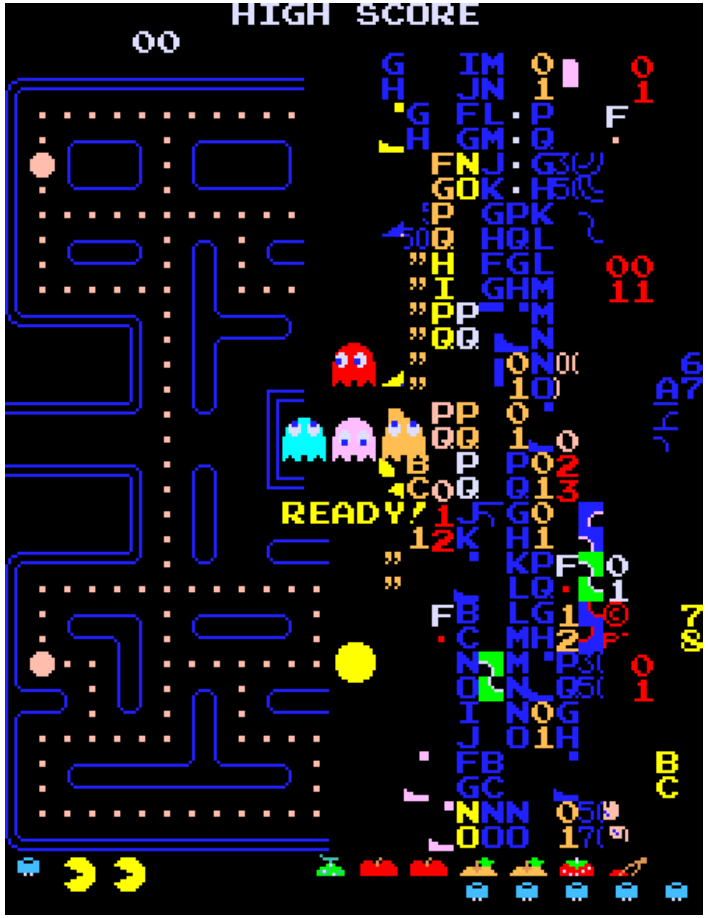
- ***Strategies for Fixing?***
  - Anticipate
  - Reproduce
    - *Things get terribly difficult if randomness is involved!*
  - Localize
  - Use proper debugging tools

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



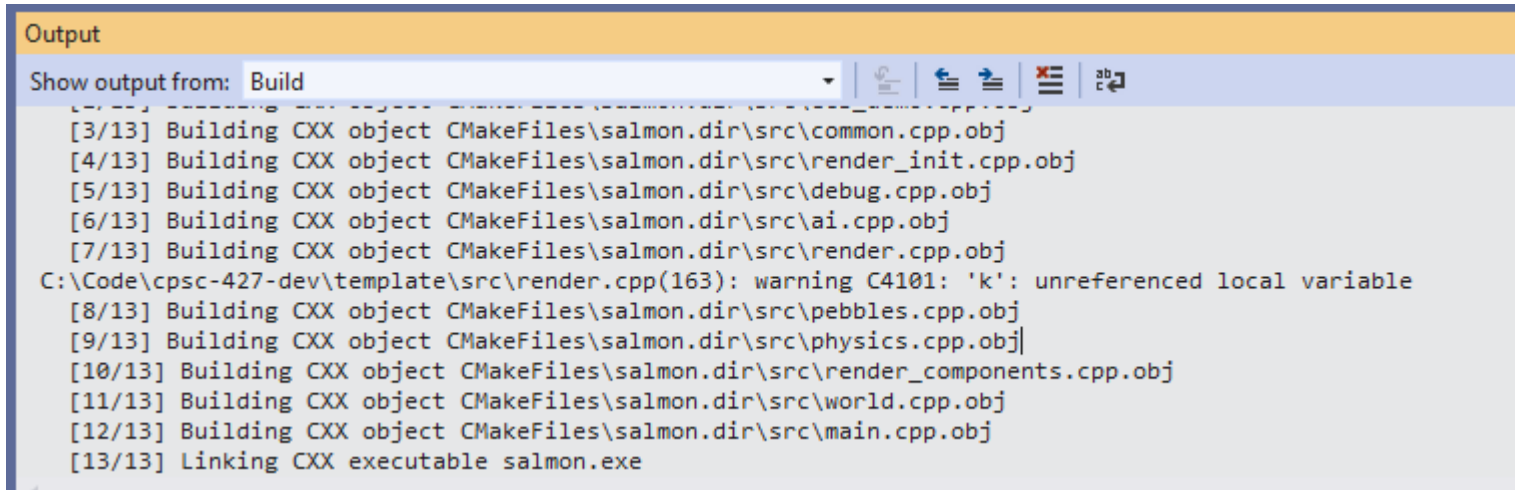
PacMan at level 256

Debugging: Strategies for Fixing?

- Anticipate I
  - *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***

Breakout rooms 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

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
- Exchange ideas on piazza

Debugging

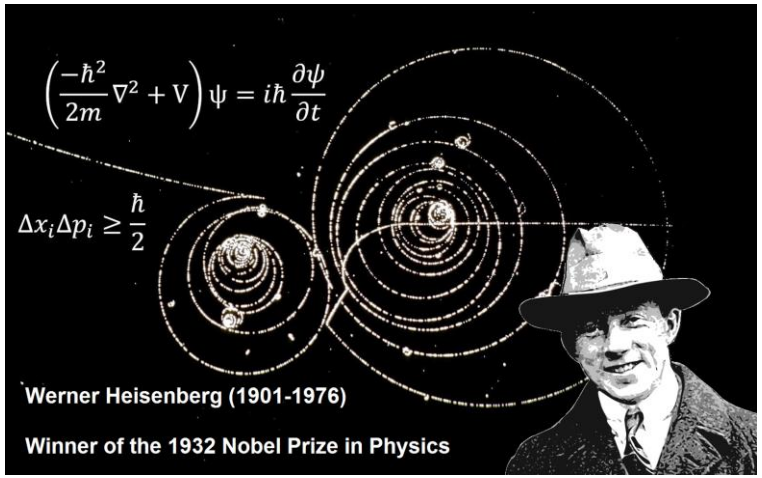
More (Human Factor) Strategies

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

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



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