Particle Review

Optimized C++

Ed Keenan

3 November 2015

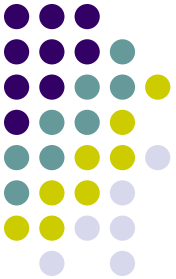
13.0.2.16.7 Mayan Long Count

Goals

- Particle system review
  - What it does
  - How to Use it
- Explain areas of interest
- Optimization ideas
- Project Deliverables
  - Competition times
  - Logs
- Contest
  - Compare on the same machine.

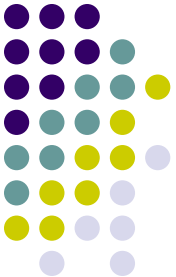
Who will win?

Keep a log



- Logs are Good!
  - Keep a work log of the tasks your are doing.
  - It’s easy to track your progress
  - Leaves you notes on what you did and how much it helped
  - Leaves bread crumbs for partial credit. ☺
- Logs are Easy to do!
  - Open up a text file or word doc and go.
    - Changelist anyone???
  - Updates take 1-3 minutes
  - Do them after each major refactor
  - Do them at the end of the day

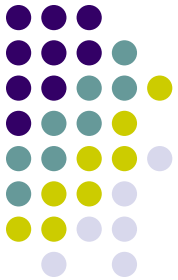
Keep a log



- Changelist: 90863 (Saturday Nov 5) *date stored in changelist automatically*
  - Added const to Vect4D
    - Saved 0.25 ms in release
  - Reworked Matrix to use references instead of pointer
    - Saved 0.05 ms in release
  - Added += operator in Vect4D
    - Reworked code to use this everywhere
    - Save 0.1 ms in release

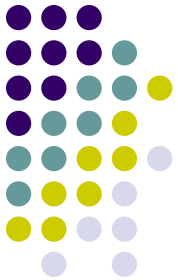
- Changelist: 90872 (Monday Nov 12)
  - Added SIMD to Vect4D
    - Saved 2.5 ms
  - Cursed Keenan's Name out loud
    - His comments in code were wrong
      - Chased a red herring for 4 hours
      - I'll get him
  - I love Linker errors
    - Sutter come-on give a little more help.

General Ideas…



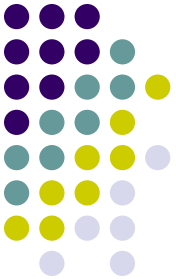
- Make constant
  - Add ***const*** everywhere
    - Function parameters
    - Constant methods
- Convert pointers to ***references***
  - Gives compiler more options
  - Removes pointer safety checks
- Remove Temporaries
  - Use **+=, -=, *=, /=** operators
  - Remember they remove temps

Things to look for…



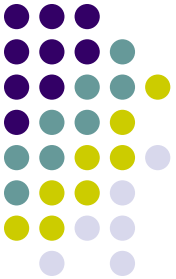
- Look for invariants
  - Stuff that doesn’t change from loop to loop
    - Remove it
- Look for useless work
  - Remove it
- Dynamic memory timings
  - Is it worth replacing

Containers and Operators



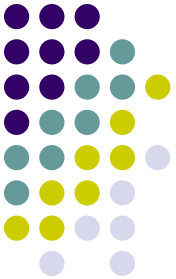
- STL containers
  - Vector
    - Is that the best container
    - Do you need containers
    - Are they being used correctly
- Must overload operators
  - Default constructor
  - Copy Constructor
  - Assignment operator
  - Destructor

Time large sections



- Time groups and parts of the program
  - Leave Timers in while you develop
- Use #if to enable multiple metrics at a time
  - Control groups of timers with a switch
- Everything changes when you refactor a section.
  - There are side effects.
- Let the data drive you
  - Not your gut feeling or what you read.
  - Timers are KING

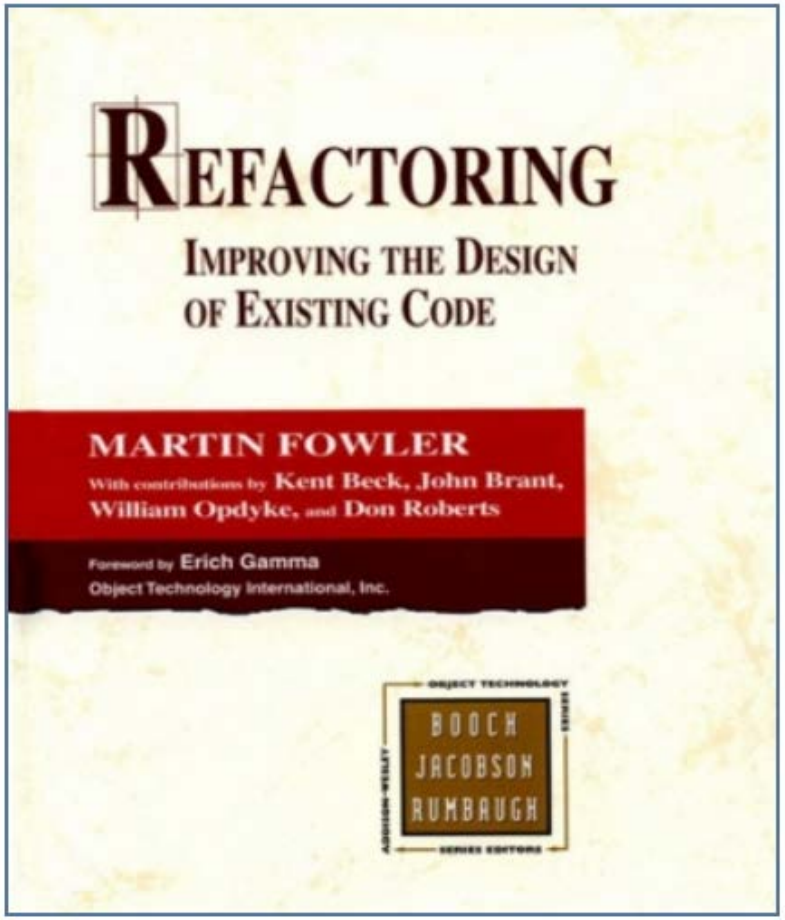
Guidelines



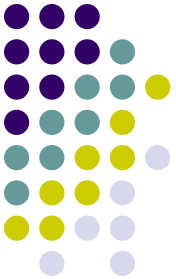
- Level Warning 4
  - Compiles cleanly in Debug and Release
- Timing
  - Need timing metrics for both configurations
  - Need original timing before any modifications
    - Turn off all extra programs on your PC when timing.
- Back up often with notes
  - Use version control
- Logs
  - Turn in Logs every week

Refactoring

- What is it?
- Who has experience with it?
- Do you really understand the principles?

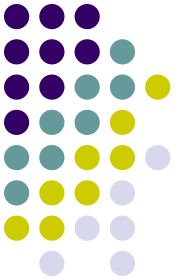


Refactoring



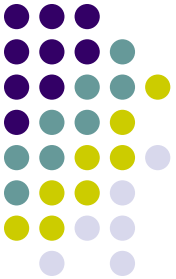
- Definition:
  - *Process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure*.
- Improving a design after it’s written

What do you Refactor



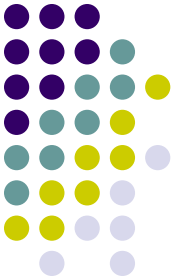
- For Maintenance reasons
  - Support
  - Development
  - Enhancements
- Architecture Improvements
- Optimizations

Why should you Refactor



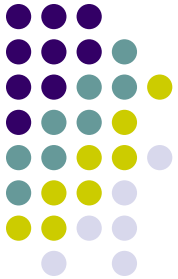
- Improves the design of software
- Makes software easier to understand
- Helps you find bugs
- Help you program faster

Rules of the road



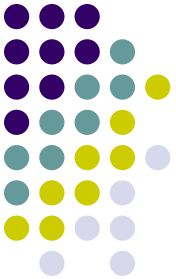
- Refactoring changes the programs in small steps.
  - If you make a mistake, its easy to find the bug.
- When you add a feature,
  - Refactor program to make it easier to add the feature.
- Testing is key
  - Make sure the system is understood and well tested, BEFORE you refactor

Important



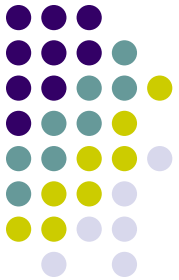
- Any fool can write code….
  - Good programmers write code that humans understand.

Rule of three



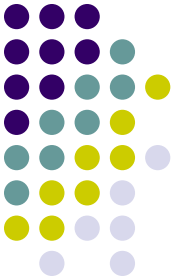
- Three strikes and you refactor
  - Refactor when you add function
  - Refactor when you need to fix a bug
  - Refactor as you do a code review

Bad Smells in Code



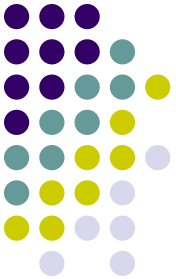
- If it stinks, Change it.
  - - Grandma Beck,
    - Discussing child-rearing philosophy

Common Smells



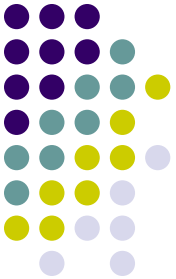
- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- Switch Statements
- Parallel Inheritance Hierarchies
- Lazy Class
- Speculative Generality

More smells



- Temporary Field
- Message Chains
- Middle Man
- Inappropriate Intimacy
- Alt classes with different interfaces
- Incomplete library class
- Data class
- Refused Bequest
- Comments

Tests



- Yes
- Have some…

Thank You!

- Questions?