Object-Level Recombination of Commodity Applications

Carleton
UNIVERSITY

Canada's Capital University

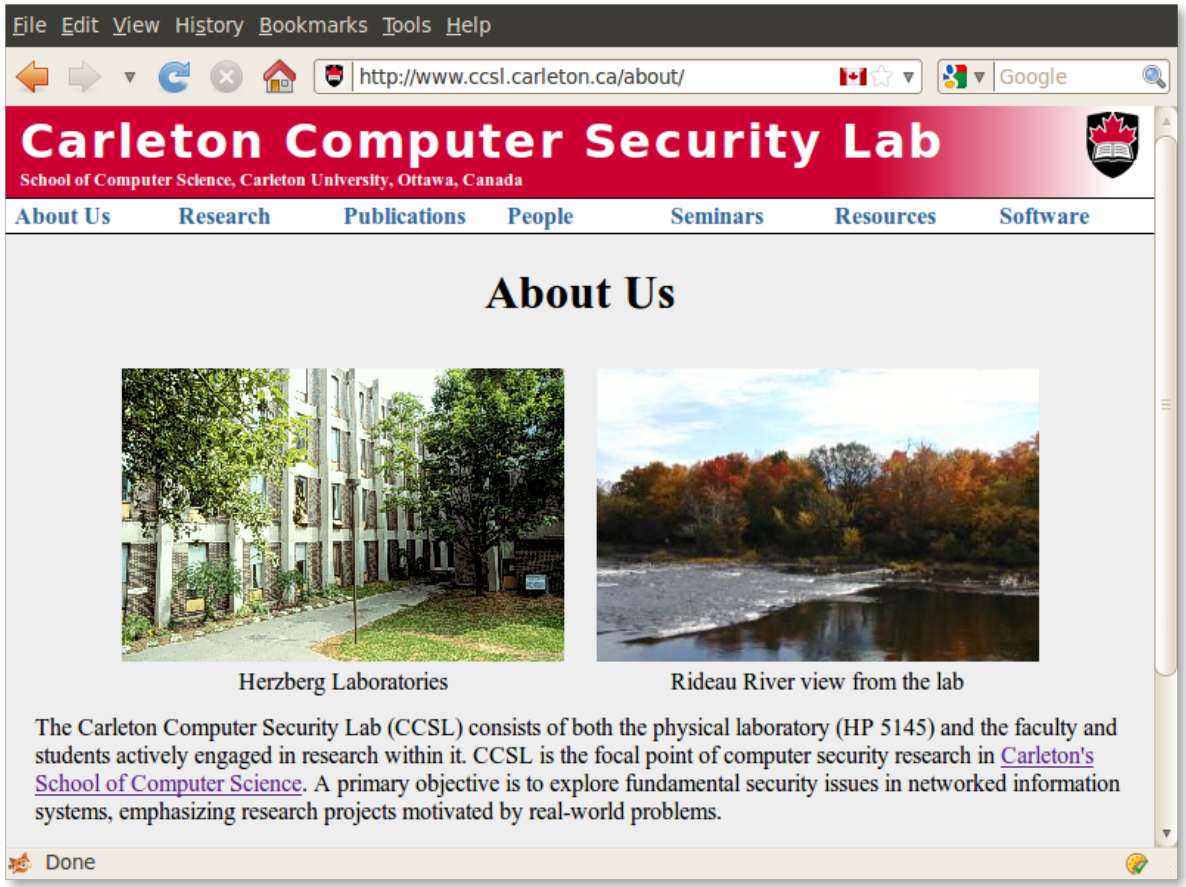
Blair Foster **Anil Somayaji**

Carleton Computer Security Lab
School of Computer Science
Carleton University

July 10, 2010

The Dream

Wouldn't it be nice to evolve **real programs** automatically?



The Reality of "Real Programs"

- **big**
  - millions of lines of code!
- brittle representations
  - C statements versus S-expressions
- brittle semantics
  - complex algorithms and data structures
- fitness functions?!

Hope

Commodity applications—the programs we use every day—can be recombined *with members of their own species.*

Huh?

Species

$$\text{Horse} \times \text{Camel} = \emptyset \Rightarrow$$

$$\text{Mozilla Firefox} \times \text{Google Chrome} = \emptyset$$

$$\text{Firefox 3.6.6} \times \text{Firefox 4.0b1} = ???$$

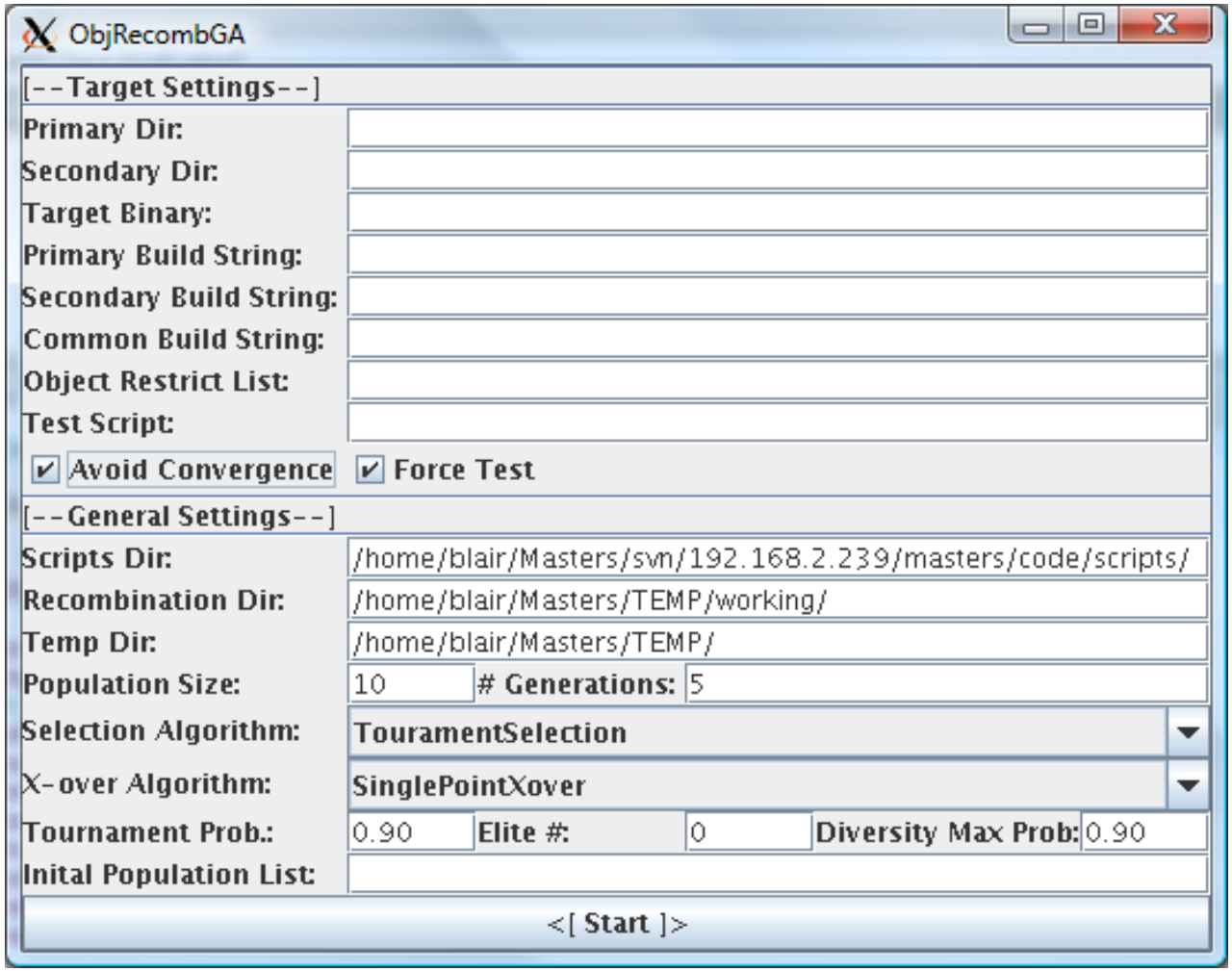
Mapping Programs to Individuals

executable	$\rightarrow$	individual
.o file	$\rightarrow$	chromosome
Linking	$\rightarrow$	development

Our Contribution

- Recombine program variants represented as object files.
- Parents must share a common recent ancestor.
- Use linker tricks, GA search to find and create viable children.
- Use a weak fitness function that only tests for basic viability; humans choose interesting variants.

ObjRecombGA

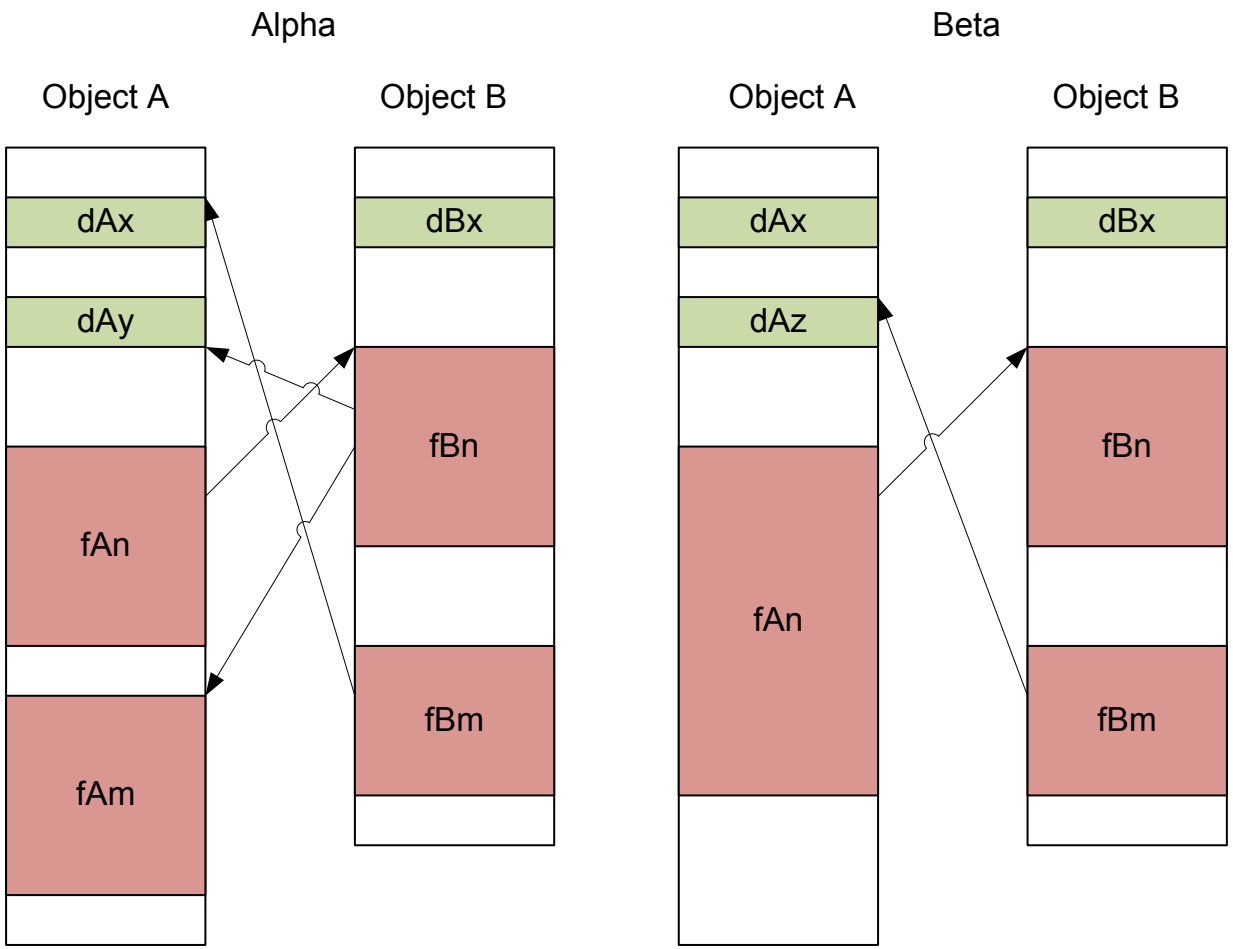


Interesting GA characteristics

- Bitstring representation: each bit selects an object file
- Small populations: 12-50 individuals
- Few generations: 20 or less
- Fast convergence to maximum fitness
  - Coarse fitness function that gives points for compiling, not crashing, and a few basic functionality tests.

Remember, this GA is designed to explore, not optimize!

The Linking Problem



The Linking Solution

- Allow linking against all object files: bitstring indicates linking *priority*.
- Do our own symbol resolution to choose appropriate variant:
  - Catalog symbols and precompute potential dependency chains.
  - Mangle all symbols to make them unique, remember mappings.
  - Rewrite symbol references to refer to appropriate version of code or data.
- And, search for viable offspring using a GA!

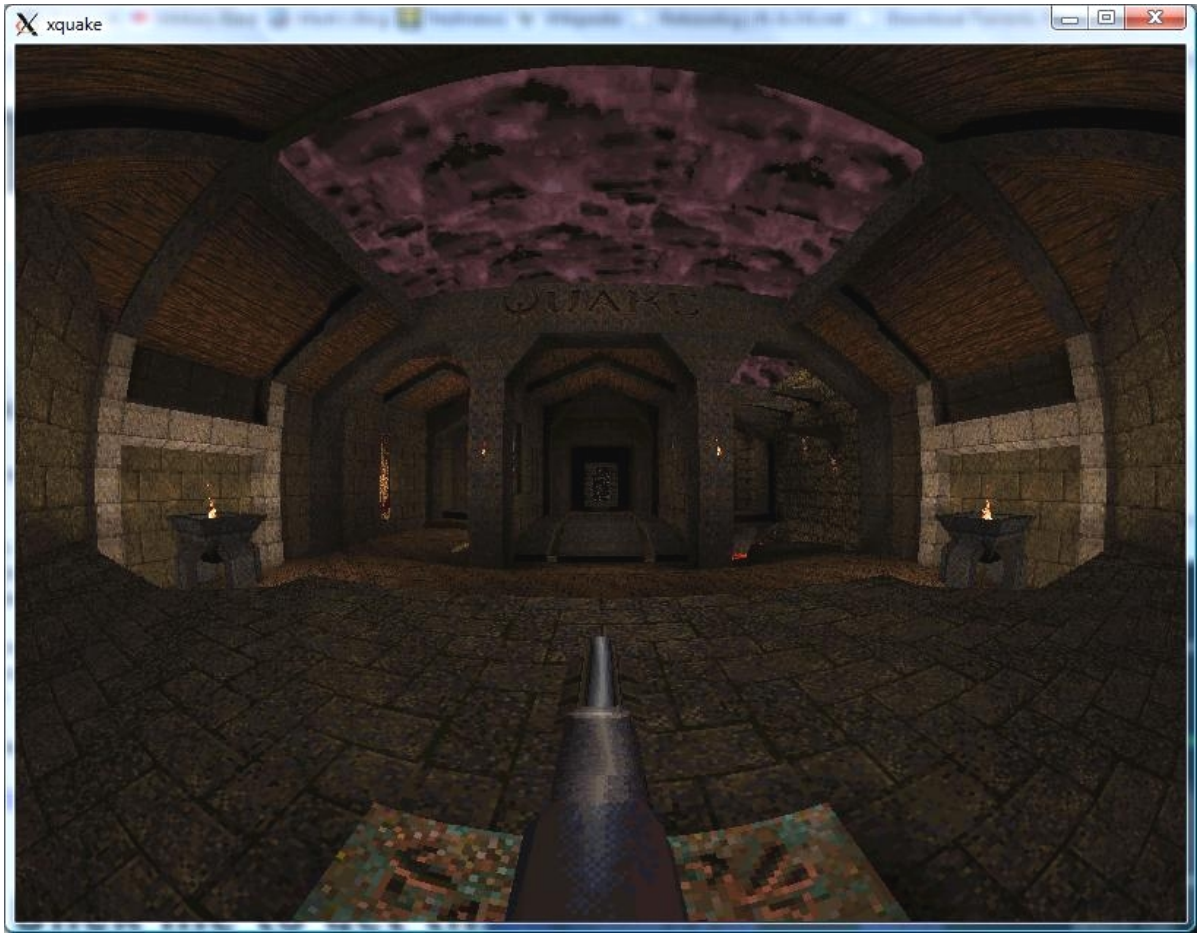
Dillo


Dillo: Dillo Web Browser :: Home Page
http://www.dillo.org
13 of 14
10.5 Kb
Dillo
8 years!
Home
Current Plans
Changelog
Screenshots
Download
Mailing List
FAQ
Compatibility
Donate
Fosdem 2005
LSM 2005
Bug Track Engine
Bug Track Intro
View Entries
Bug Insertion
Volunteering
Developers
New Developer
Naming&Coding
Dpi1 Spec
CSS Spec
Source repository
Authors
Security contact
Users
Help
Icons
Bug meter
The Dillo Web Browser
Welcome to the Dillo Project!
What is Dillo?
Dillo is a multi-platform graphical web browser known for its speed and small size.
Dillo is written in C and C++.
Dillo is based on FLTK2, the Fast Light Toolkit (statically-linked by default!).
Dillo is free software made available under the terms of the GNU General Public License (GPLv3).
Dillo strives to be friendly both to users and developers.
Dillo helps web authors to comply with web standards by using the bug meter.
Project Objectives:
The democratization of internet information access.
Personal security and privacy.
High software efficiency.
So, you have been using dillo for a while, but have you read about the project objectives? Please give it some time, you'll find some interesting issues there!

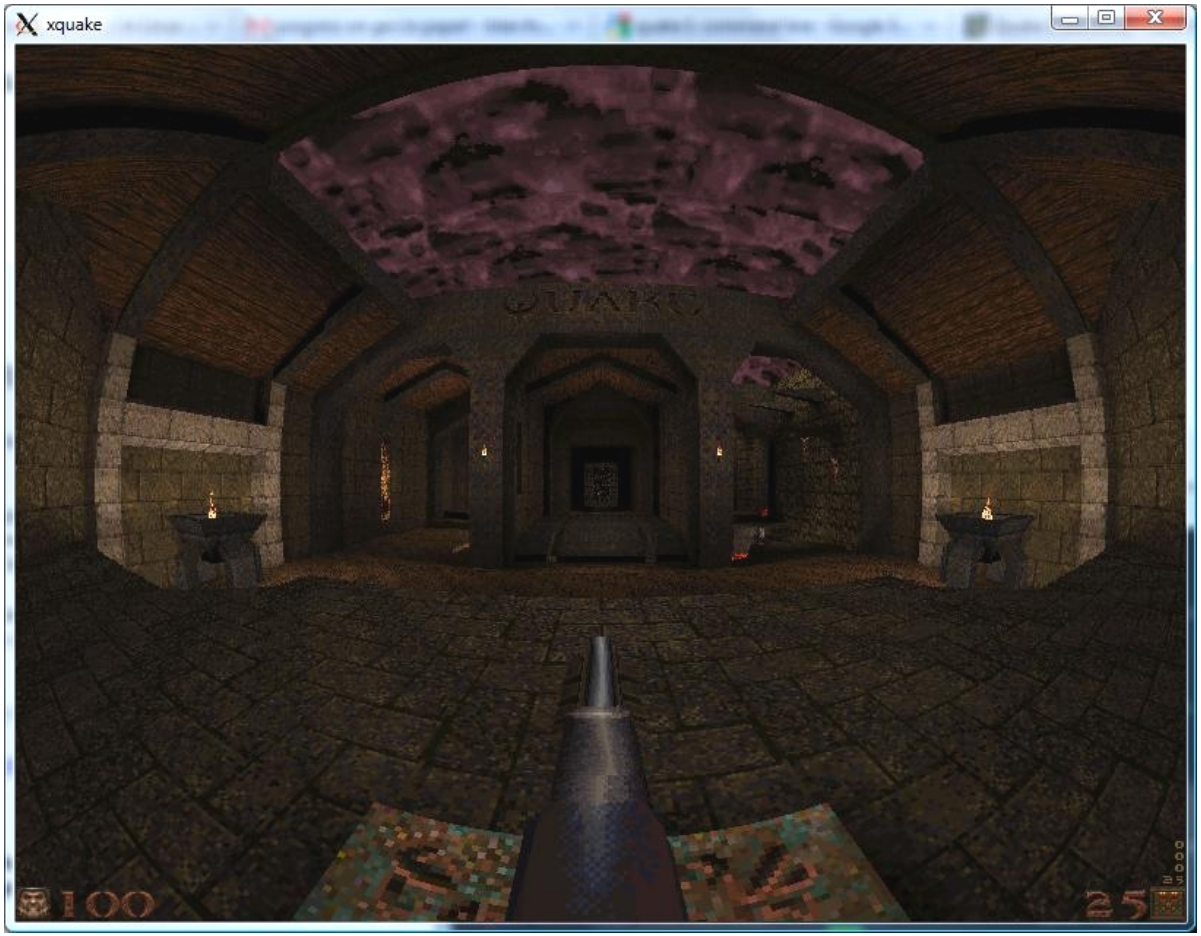

Dillo: Dillo Web Browser :: Home Page
Dillo
8 years!
Home
Current Plans
Changelog
Screenshots
Download
Mailing List
FAQ
Compatibility
Donate
Fosdem 2005
LSM 2005
Bug Track Engine
Bug Track Intro
View Entries
Bug Insertion
Volunteering
Developers
New Developer
Naming&Coding
Dpi1 Spec
CSS Spec
Source repository
Authors
Security contact
Users
Help
Icons
Bug meter
Related
Dillo in the Press!
Interview
The Dillo Web Browser
Welcome to the Dillo Project!
What is Dillo?
Project Objectives:
So, you have been using dillo for a while, but have you read about the project objectives? Please give it some time, you'll find some interesting issues there!
Current version
Dillo follows an evolving software model where each new version should be better than the previous one; there's no place for unstable releases, so just keep with the latest one: dillo-2.0.
Click here for the downloads page.
Code contributions

Quake: HUD, no models

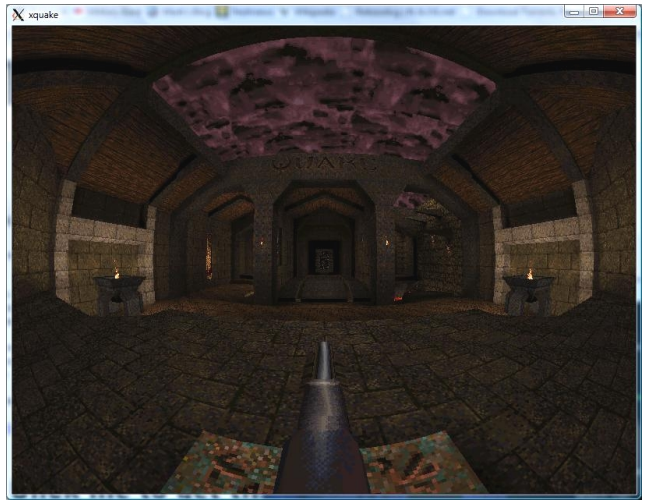
Quake: Fisheye

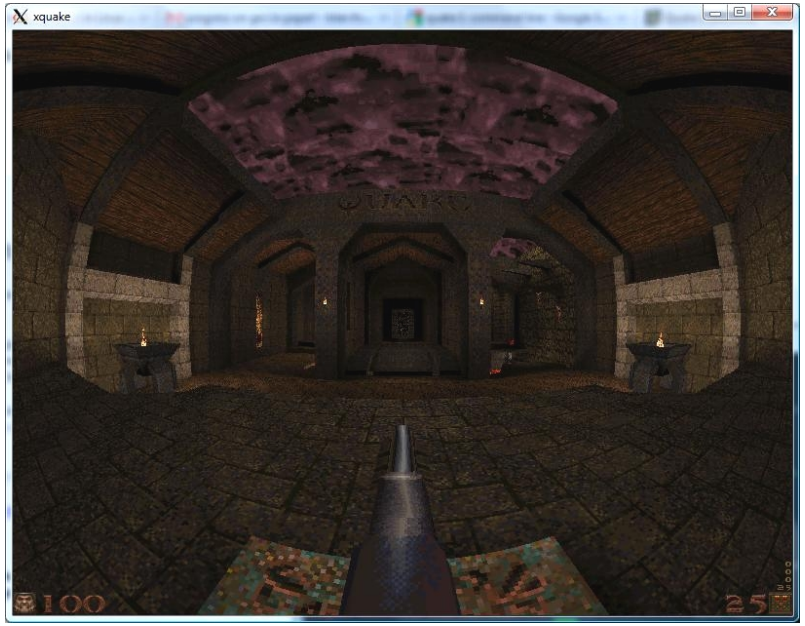


Quake: Recombined Child

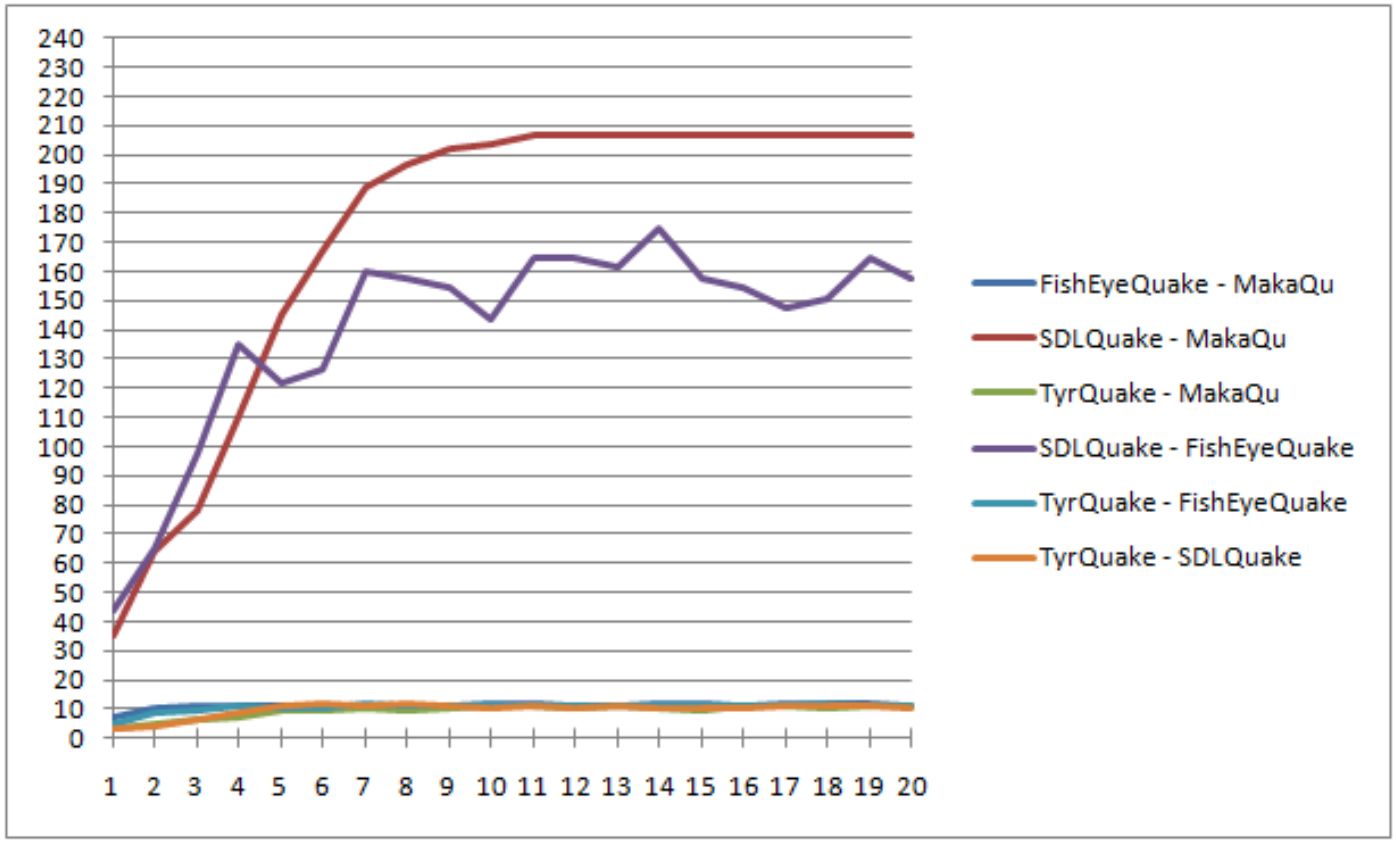


Quake: Parents and Children





Fitness Convergence



Note that populations exhibit high diversity.

Limitation of current implementation

- Two ancestors only.
- Child programs cannot be used as ancestors
  - Children are effectively diploid, parents are haploid.
- Linking sometimes fails when it could have succeed.
- Primitive fitness functions.

Potential Applications

- Software testing/debugging
- Functional diversity for security
- User-directed software evolution

Conclusion

- Commodity software can be recombined using object files as the units of recombination.
- Linking problems are significant but can be overcome with linker tricks.
- Functional dependencies are addressed through GA search.
- Only works with closely related programs.
- Defines a "species" relationship amongst programs.
- Opens the door to automated evolution of the software we use everyday.

Acknowledgements

- Funded by Canada's NSERC Discovery Program
- Blair Foster (Master's student!)
- "Road to Software Evolvability," Santa Fe Institute 2005
- Stephanie Forrest (graduate advisor), David Ackley

Contact me at soma@ccsl.carleton.ca

Questions?