jspatial

http://io7m.com/software/jspatial

code@io7m.com

Why?

Why?

Computers lack spatial awareness!

Example: For efficiency reasons, when implementing collision detection for physical simulations, we want to limit comparisons between pairs of bodies to those that are in close proximity to each other.

By subdividing the world into small spaces, and only comparing the pairs of bodies within those spaces, we reduce the number of necessary comparisons by orders of magnitude.

Trees

Trees

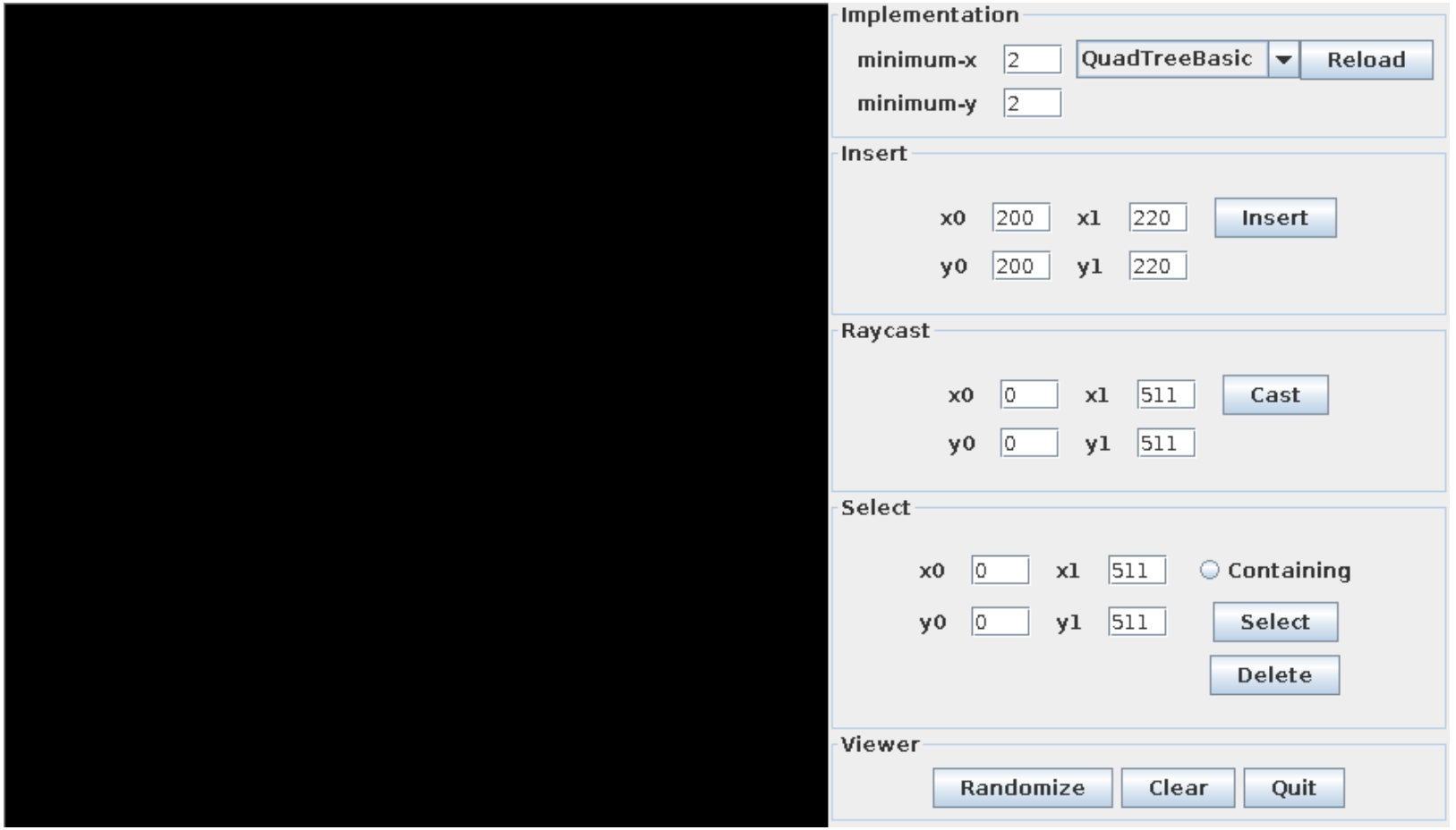
The **jspatial** package offers multiple implementations of **quadtree** and **octtree** data structures.

Quadtrees

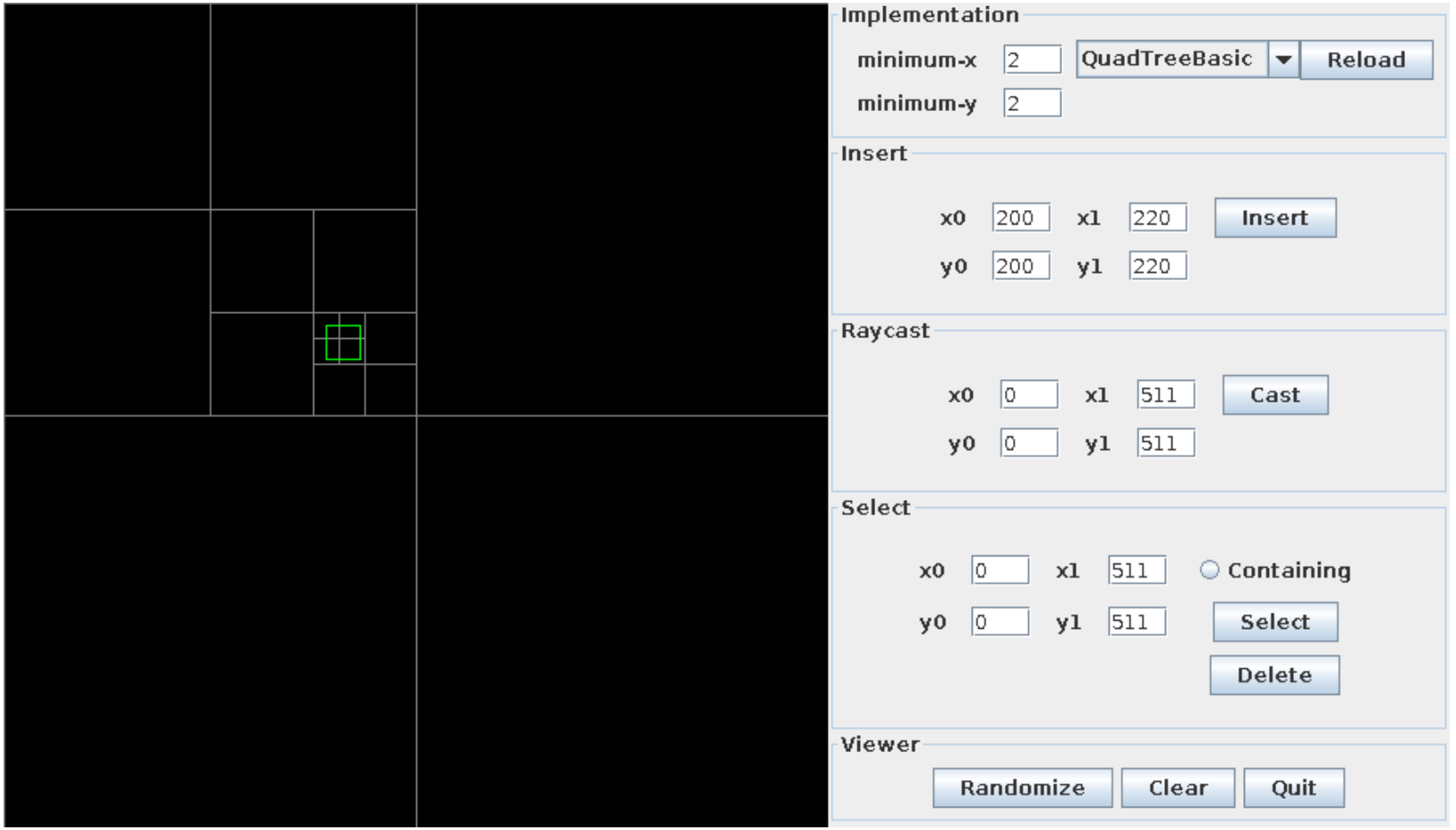
Quadtrees recursively divide a 2D space into quadrants.

A **quadtree** initially contains one cell and is divided into smaller cells upon insertion of an arbitrarily-sized object.

Quadtrees

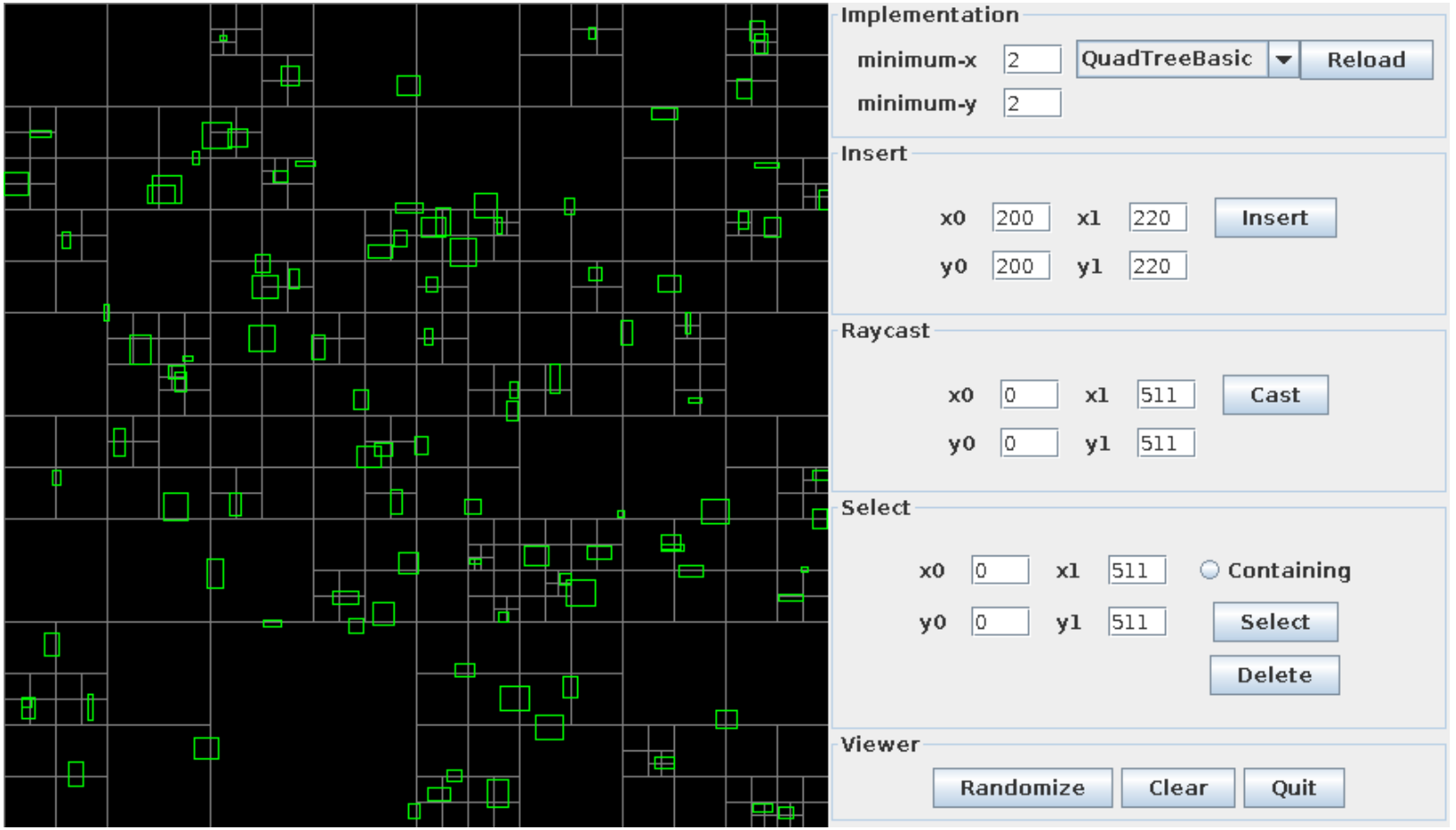

Implementation
minimum-x
2
QuadTreeBasic
Reload
minimum-y
2
Insert
x0
200
x1
220
Insert
y0
200
y1
220
Raycast
x0
0
x1
511
Cast
y0
0
y1
511
Select
x0
0
x1
511
Containing
y0
0
y1
511
Select
Delete
Viewer
Randomize
Clear
Quit

Quadtrees



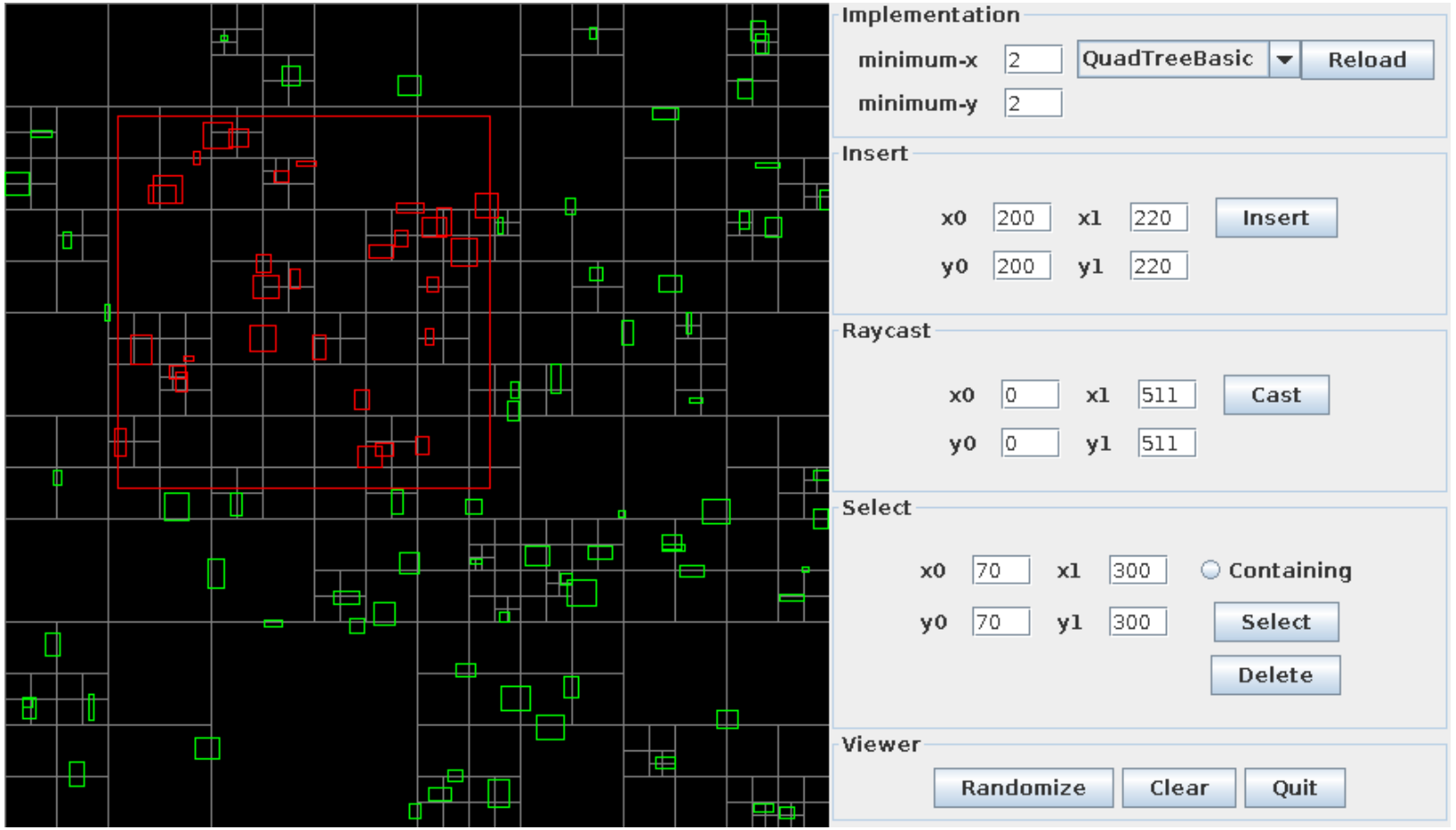
Note how the space has been divided into quads...

Quadtrees



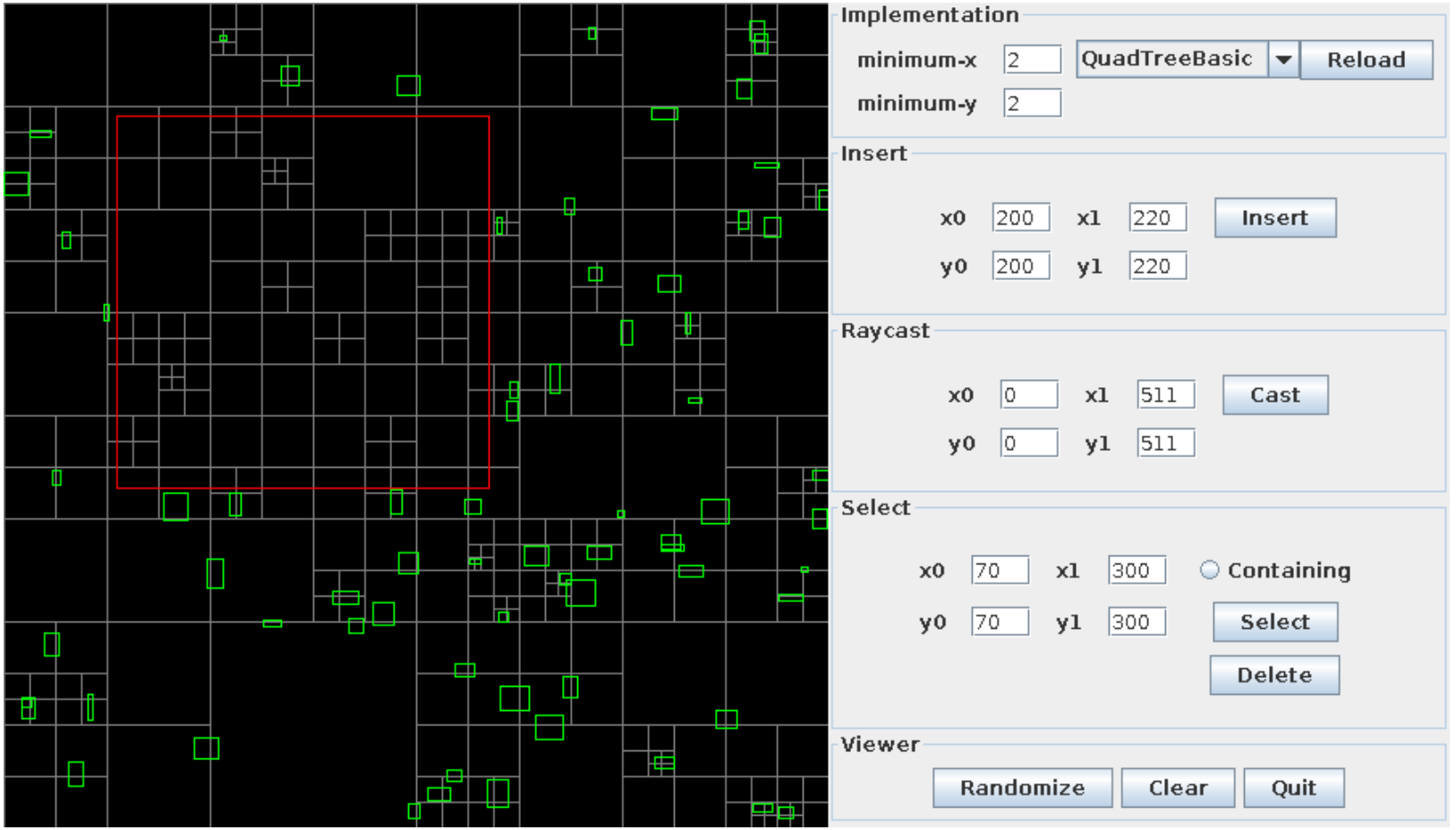
Object insertion is fast and requires no preprocessing

Quadtrees



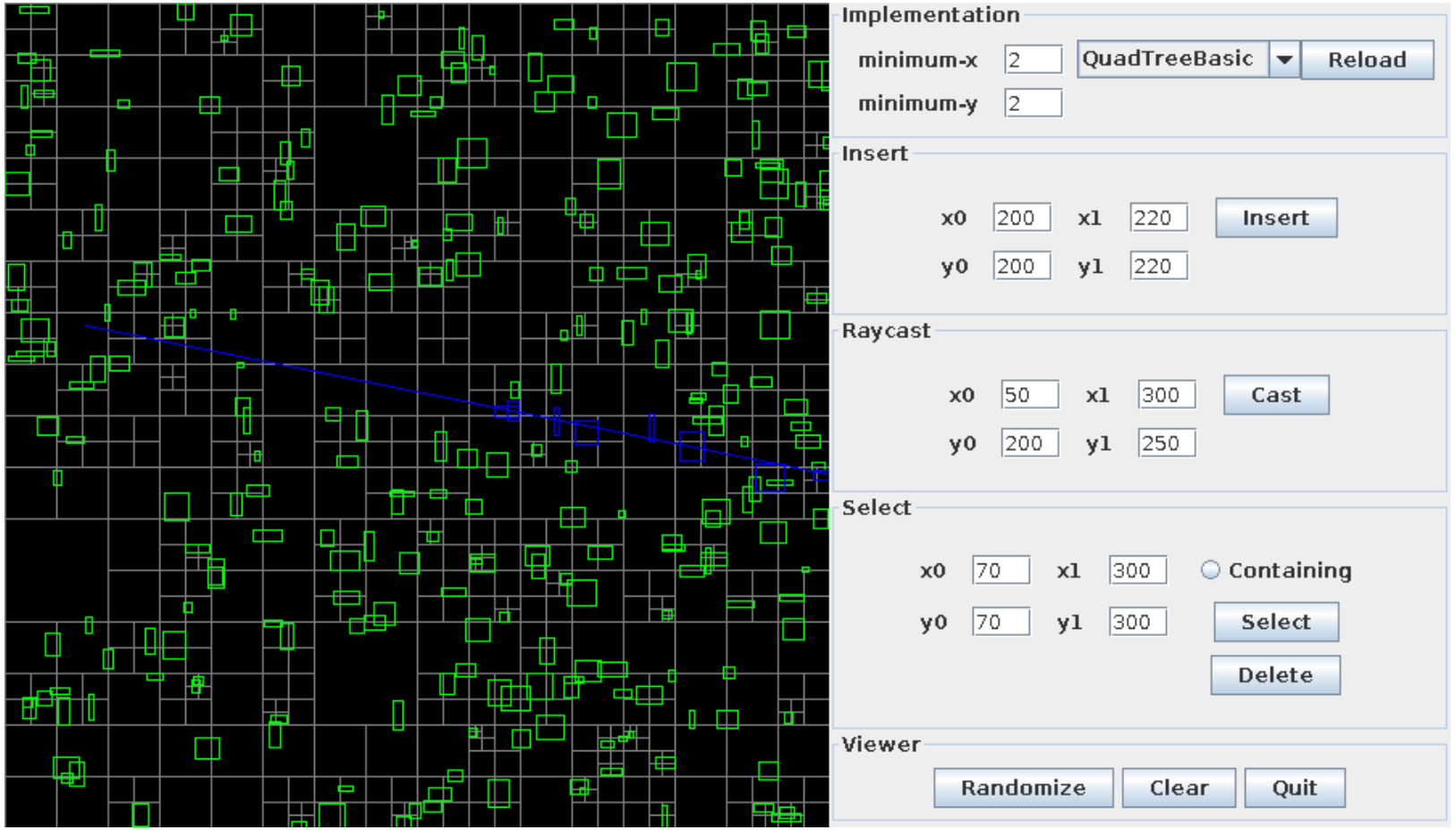
Determining which objects are contained or overlapped by an area is efficient

Quadtrees



Objects may be removed from the tree efficiently

Quadtrees

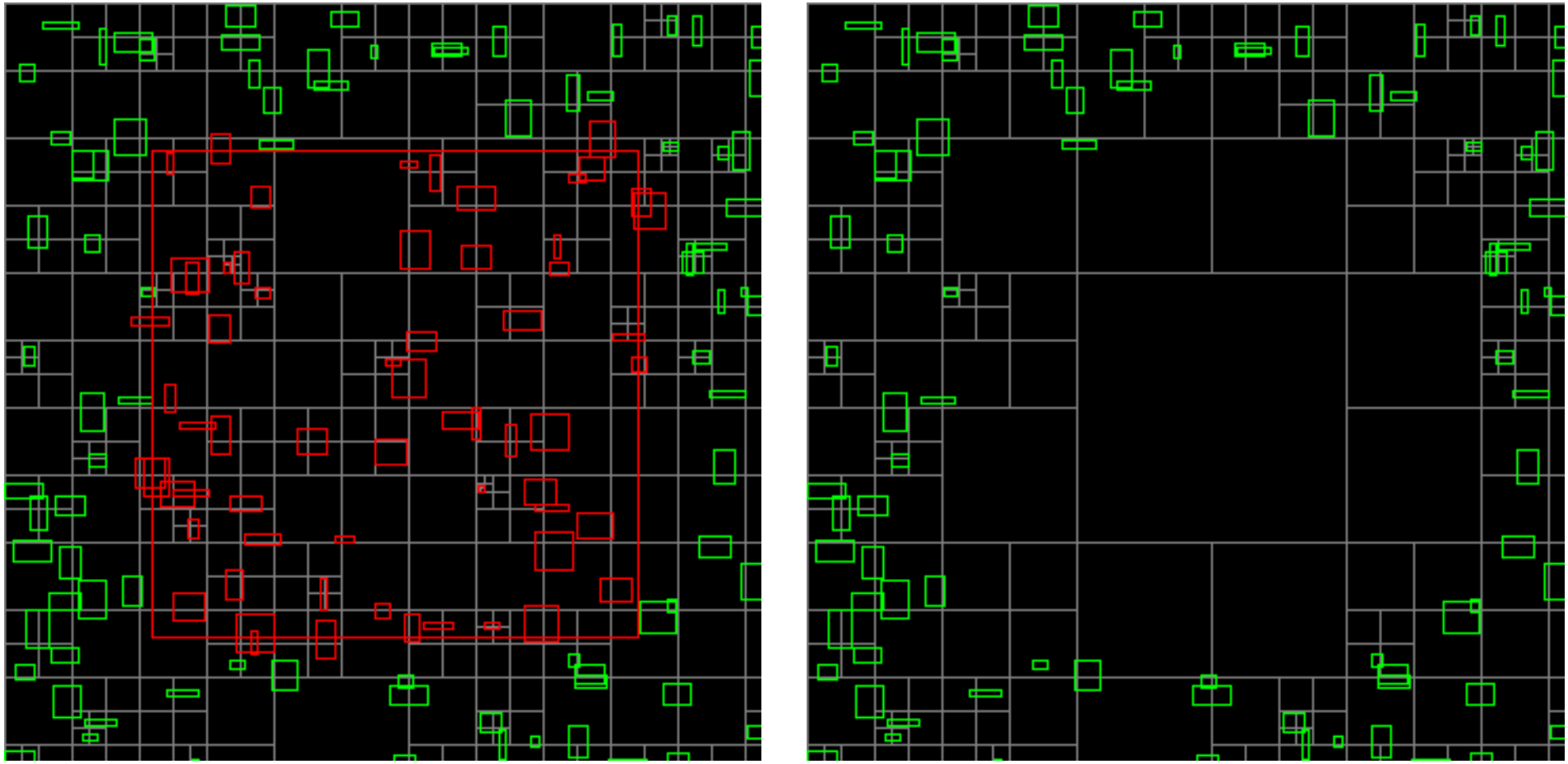


Ray casting queries are supported

Quadtrees

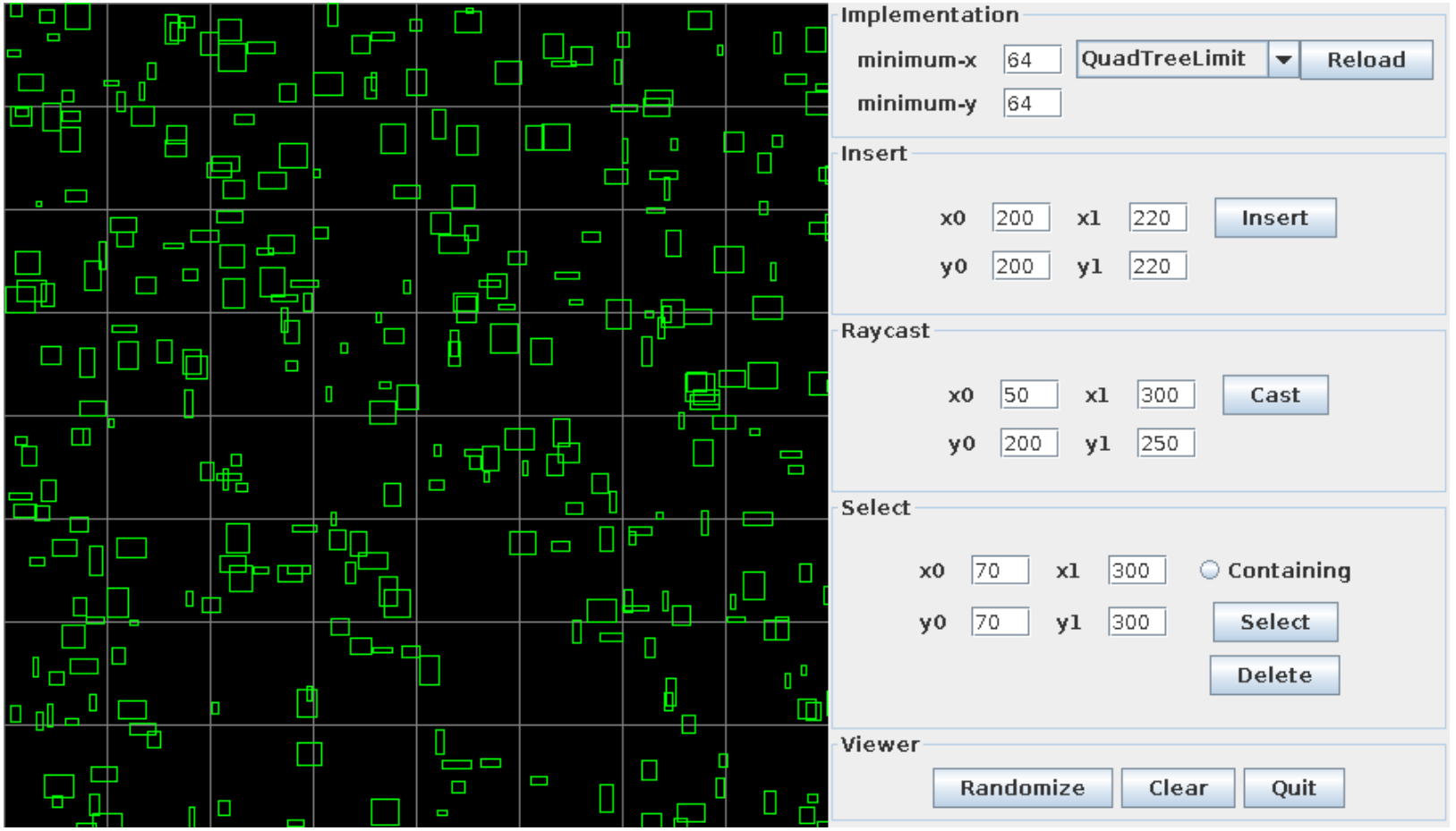
Multiple variations on the basic **quadtree** algorithm are included.

QuadTreePrune



The **QuadTreePrune** algorithm aggressively removes empty tree cells

QuadTreeLimit



The **QuadTreeLimit** algorithm places a lower bound on the size of cells

Quadtrees

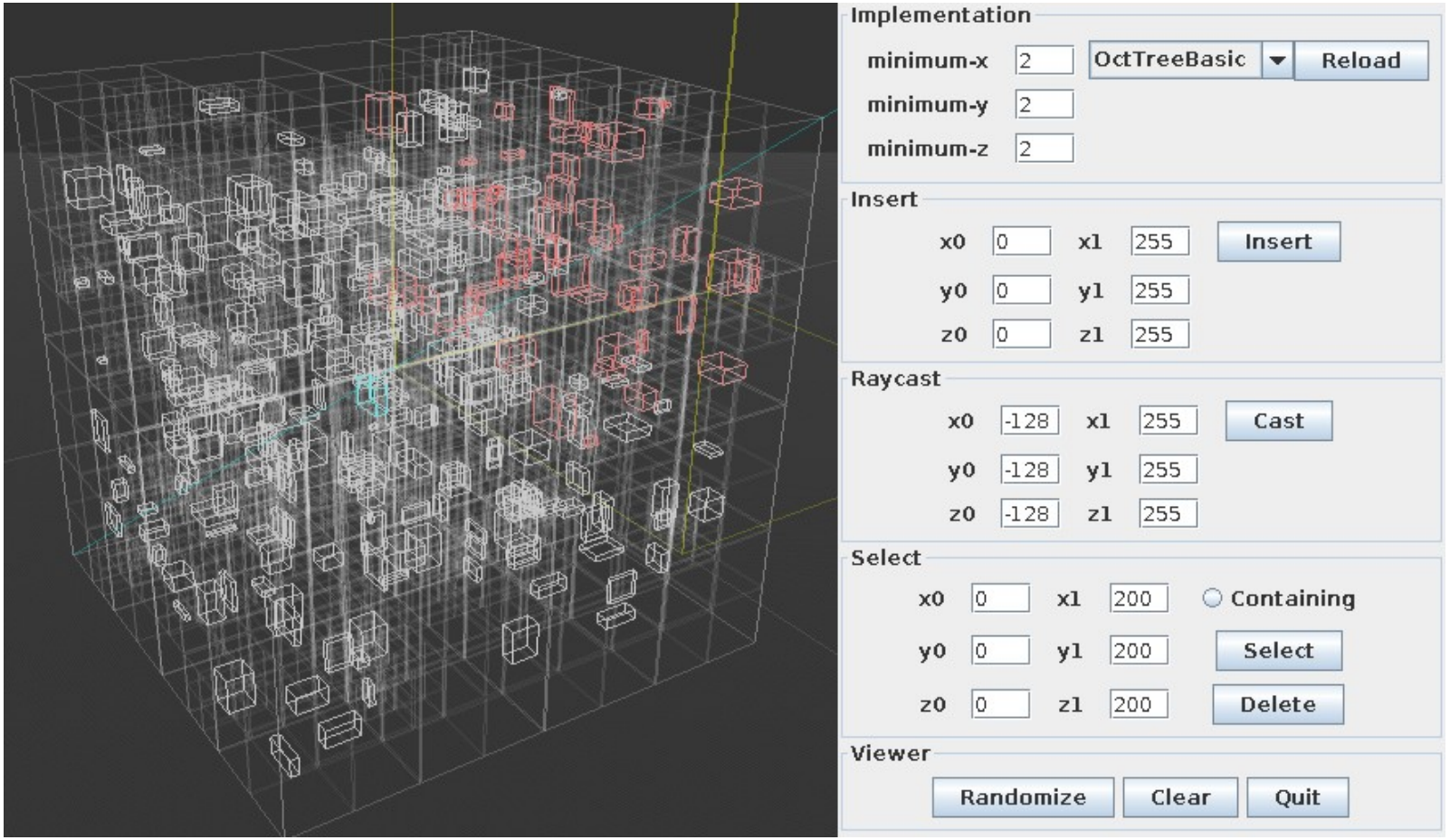
The **jspatial** package does not pretend to know which algorithm is right for you.

The implementations share common interface types, meaning that developers can choose from any of the implementations without needing to modify code.

Octtrees

Interestingly, the **Quadtree** algorithms generalize to three dimensions.

Octtrees



All of the previous queries are supported in three dimensions

Octtrees

As with **quadtrees**, the **jspatial** package does not pretend to know which algorithm is right for you.

The implementations share common interface types, meaning that developers can choose from any of the implementations without needing to modify code.

Deja vu!

Usage

Usage

Have your object implement either of the **QuadTreeMember** or **OctTreeMember** interfaces (or both, if you're that way inclined).

The interfaces consist of a pair of trivial functions that request the axis-aligned bounds of the object.

Usage

Initialize a **quadtree** or **octtree** using the available constructors.

Usage

Insert your objects into a tree.

Usage

Query the tree.

Usage

You're done!

Usage

The package comes with a tutorial, API documentation, and some example viewer programs. The viewers have been used to provide the screenshots in this presentation.

The example viewers use Swing and JOGL (http://jogamp.org/) for rendering, but the **jspatial** package itself is completely renderering-system-agnostic.

Yes, the package works on Android!

Engineering

Engineering

Of course, all of the features of **jspatial** package would be meaningless if developers could not use the software due to poor engineering practices on behalf of the author.

So, **jspatial** works well because...

Engineering

Minimalism

Packages on **io7m.com** strive to do one thing well, and nothing more.

Engineering

Type safety

All packages on **io7m.com** strive to use the type-checker to eliminate as many classes of bugs as possible, and to enforce program invariants statically.

The **jspatial** package is no exception.

Engineering

Ease of use

All packages on **io7m.com** strive to be easy to use in terms of APIs. Interfaces are designed to statically disallow incorrect usage.

The **jspatial** package is no exception.

Engineering

Well-documented

All packages on **io7m.com** come with complete documentation, including tutorials and generated JavaDoc.

The **jspatial** package is no exception.

Engineering

Well-tested

All packages on **io7m.com** come with extensive and exhaustive batteries of unit tests.

The **jspatial** package is no exception.

(Currently, the package has 96.5% test coverage of all instructions and branches).

Engineering

Well-versioned

All packages on **io7m.com** use semantic versioning (http://semver.org/), meaning that you can trust that APIs will not change underneath your application without warning, and that all backwards-incompatible changes are clearly indicated.

The **jspatial** package is no exception.

Engineering

Well-supported

The author offers direct support from #io7m on Freenode (http://freenode.net/).

Please note that I do need to sleep occasionally. Be patient!

Engineering

Permissively licensed

All packages on io7m.com are licensed under the ISC license – the shortest and most liberal BSD-type license in current use. Use in closed and/or commercial programs is permitted and encouraged.

The jspatial package is no exception.

Engineering

Easily deployed

All packages on **io7m.com** are deployed to Maven Central (http://search.maven.org/) and can therefore be used instantly with any build system that uses Maven packages.

Compiled packages, source code, and documentation are also available from their respective project pages on **io7m.com**.

Source code is managed by the Fossil SCM (http://fossil-scm.org/).

Thanks!

Mark Raynsford
http://io7m.com/
code@io7m.com

PGP: 0x60824220