Spatial Indexes for Inkscape

Special Course, Spring 2009
Technical University of Denmark

Project funded by ELLAK (http://www.ellak.gr/)

Evangelos Katsikaros
DTU supervisor: François Anton
Inkscape mentoring: Nathan Hurst
ELLAK supervisor: Diomidis Spinellis

June 6, 2009

Contents

Acknowledgements

This project is a multidisciplinary effort and I would personally like to thank Nathan Hurst for mentoring and helping me smoothly dive in lib2geom's code, François Anton who made this project a special course for my master's curriculum at DTU, Diomidis Spinellis for coordinating the project with ELLAK and ELLAK for funding the project.

1 Introduction

This report summarizes the project implemented during the special course "Spatial Indexes for Inkscape", during spring 2009, at the Technical University of Denmark. This special course is mainly defined by the project delivered at the end. Its purpose is the understanding of the theoretical background of spatial indexes, as well as gaining on-hands experience by implementing R-Trees, for "Inkscape" a popular open source vector graphics editor.

The implementation is also the deliverable for an implementation contest organized by ELLAK [1]. The goal of the contest to promote open source software and implement software useful for business and education purposes.

In this first section of the report, we introduce the topic of the project, the purpose and its goals. In section 2 we present the process of implementing the project, and in section 3 we show two demo applications. Finally, we describe the project's deliverables in section 4 and we summarize in section 5.

1.1 Purpose

Inkscape is a vector graphics editor, similar to Adobe's Illustrator and Corel's CorelDraw, that uses SVG as its native format. The application is open source, licenced under GNU GPL version 2 [3]. The developers of Inkscape have created a separate library, called lib2geom, where they intend to migrate all the 2D functionality that could be needed by the application. These include native point, rectangular and other shape classes, calculation of shape intersections, unions, calculation of path lengths and other 2D calculations (like path or arc lengths). Moreover, lib2geom offers a GUI framework, called "toy framework", that facilitates the swift creation of GUI mockups and tests when implementing functionality for the library.

During the rendering of every window update, Inkscape has to find the shapes that intersect with the visible window, so that only these are rendered. This query is performed by scanning all the available shapes of the displayed file. This process doesn't use any kind of indexing and it is a clear case where 2D spatial indexes could improve the performance. R-Trees is a very popular spatial indexing method, already successfully used

in geospatial extensions of major RDBMS systems like Oracle [7], MySQL [6], PostgreSQL [8].

Our main sources of information for this project were "Spatial Databases: A Tour" [9] and "Rtrees: Theory and Applications" [5]. The implementation of R-Trees is based on Guttman's original paper "R-Trees: A Dynamic Index Structure for Spatial Searching" [2], that introduced R-Trees. The paper fully describes the insertion, deletion and update procedures, so we are not going to repeat these here in this report. The interested reader can find the all the information needed in this paper which is also freely available.

1.2 Goals

The main goal is the implementation of an R-Tree index, more specifically the flavor described in Guttman's original paper, using quadratic split. Moreover:

- The different flavors of R-Trees use different splitting techniques, so the data structure should be implemented in a way that the implementation of additional splitting techniques is easy
- Implementation of test cases that show the proper functionality of the tree during inserting and erasing elements.
- Implementation of a GUI application, using the handy "toy framework", to visualize the way the R-Tree works. Its puprose is twofold: it was used as "visual debugger" during the implementation and it can be used as an educational tool to understand R-Trees.

2 Implementation

As we mentioned before, we implemented the R-Tree with quadratic split [2]. The operations supported by this index are search, insert, delete, update and the algorithms for each one are fully described. Their steps are not strictly defined in a pseudo language but are described in natural language.

In this section we are going to describe the implementation process and the problems we run into. In 2.1 we describe the data structure, in 2.2 the insert procedure, in 2.3 the search procedure, in 2.4 the erase procedure and finally in 2.5 how we scan all the elements of the tree.

2.1 Data Structure

The first challenge of the project was the design of a data structure able to represent in a nice way the R-Tree.

The main characteristic of R-Trees, is that the indexing of shapes is based on their bounding boxes. Each node holds a number of records. R-Trees have two types of records.

- Leaf records which contain:
  - a pointer to a shape.
  - the bounding box of the shape.
- Non-leaf records which contain:
  - a pointer to a child node (data)
  - the bounding box of the child node

So in our design, each node has the ability to store both leaf and non-leaf records, but we only use one of them according to the positioning of the node in the tree. If the number of non-leaf nodes is greater than zero then the node holds non-leaf records, and if the number of leaf nodes is greater than zero then the node holds leaf records. The UML class diagram of the above are summarized in Figure 1.

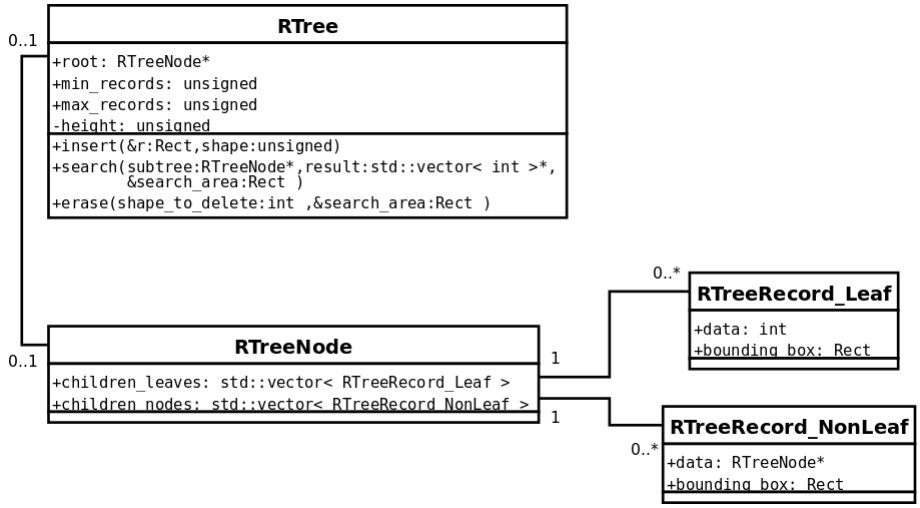


Figure 1: UML class diagram of R-Tree, its nodes and the records of each node (leaf and non-leaf). The tree keeps the root and internally also keeps track of the tree height. Each node is able to store both types of records but is making use of one of them only.

Another issue is the the two kinds of records, force the code to be duplicated in order to accommodate the read/write of the proper data type. Obviously, class member `bounding_box` has the same data type in both cases, but class member `data` can vary (pointer to shape or pointer to child node). During the operations of the tree:

1. we care whether we touch a leaf or non-leaf record, since we write data in the node, so we want to write the correct data type (shape or child node).

2. we do *not* care whether we touch a leaf or non-leaf node, because we only read/write the bounding boxes which are of the same data type in both cases.

In our design we were able to eliminate the duplication of the 2nd case, but the first couldn't be avoided.

Moreover, we don't keep the parent of each node, but we search for them whenever they are needed.

2.2 Insert

We begun with the most complex of the four operations, insert. The fact that the algorithms are described in natural language gives some freedom in the design of some steps. One such point is method "Adjust Tree", where the algorithm doesn't explain how the values of the splitted node are inserted in the existing nodes (step AT1). The paper implies that when a node is split, then one of the two new splitted nodes is copied to the existing node and the other is handled by step AT4.

Lib2geom is written in C++, so we applied techniques such as const correctness in arguments and functions, to make sure that only the appropriate functions can modify the contents of the tree.

The structure of insert operation was modified after we implemented erase operation (see 2.4).

2.3 Search

After insert, we implemented search, a straightforward procedure, without any difficulties.

2.4 Erase

Finally we implemented the erase function. Erase is not complicated, but there was one thing that delayed its implementation. During this procedure, we erase one element and if the node is underfull, we put the remaining records of the node in a set, called Q, and we propagate the changes upwards. When we reach the root, we re-insert all the elements of set Q. But, we have to make sure that each element is re-inserted at the same height as before. So, we had to modify insert in order to accommodate this change.

The final version of insert has the ability to insert a leaf record, or if a flag is set, we insert a non-leaf node at the correct height (it's given as an argument).

2.5 Tree Walk and Checks

In order to scan all the contents of the tree, we implemented B-Tree's inorder tree walk. In the case of R-Trees the ordering of the walk doesn't reveal a

concrete ordering of the records, since the records inside a single node are not ordered. But we can clearly see the structure of the whole tree.

Modifications of the tree walk are used to:

- print the contents of the tree to the standard output.
- find the parent of a node.
- perform sanity checks.

To make sure that the properties of the rtree are satisfied after operations, we check that each node always has a number of records between the minimum and the maximum allwed, and that the tree is balanced.

3 Demo Applications

In this section we are present two applications based on lib2geom's R-Trees. In 3.1 we present a GUI and in 3.2 a test application.

3.1 GUI

As we mentioned before, lib2geom offers "toy framework", rapid development GUI framework, for simple GUIs. Its main puspose is the creation of mockup applications and testing of lib2geom's functionalities. The rendering is handled by libcairo [4], a anti-aliased vector-based API that outputs to X Windows, PDF, image formats, and others.

We used this framework to create `rtree-toy`, a GUI application where the user can graphically manipulate an R-Tree and see the bounding boxes of the nodes of all heights.

More specifically the application has the modes insert, delete, search, which perform the appropriate function.

- In insert mode, the user can draw a new rectangle by clicking and dragging the mouse. When the mouse is released the new rectangle is inserted in the tree.
- In search mode, the user can draw a rectangle that represents the search area. The user can see all the rectangles that intersect with the search area in the standard output.
- In delete mode, the user can click on the border of an existing rectangle and erase it from the tree.

Another interesting feature (Figures 3 to 7) is that the bounding boxes of each level of the tree can be seen separately. This allows the user, to see in action the impact of the tree's operations.

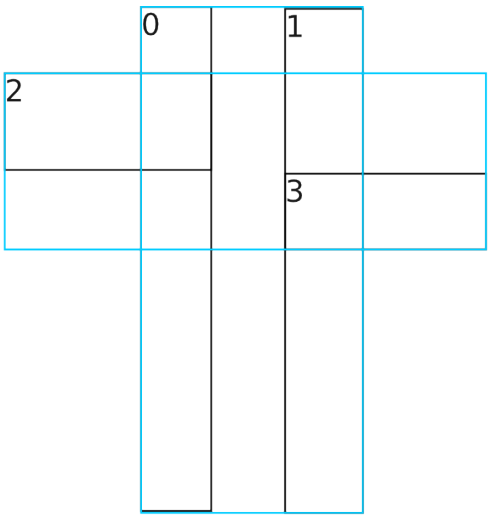


Figure 2: Screenshot from the application. The R-Tree used is a 2/3 (min records per node/max records per node) and is a representation of Guttman's "good splitting" example. The last line shows the current mode of the application. The middle line shows whether the bounding boxes are visible or not, and if all the layers of bounding boxes are visible. With black color are depicted the shapes and with cyan the tree's bounding boxes.

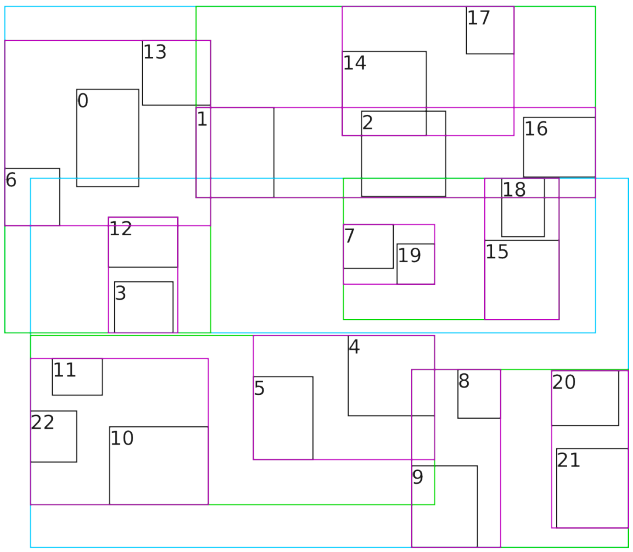
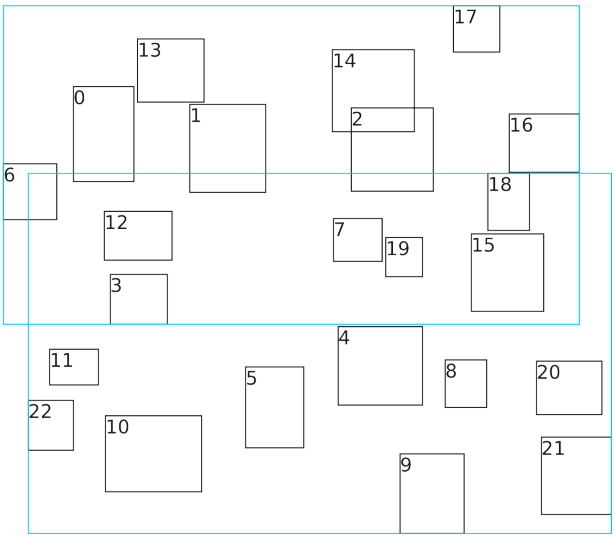
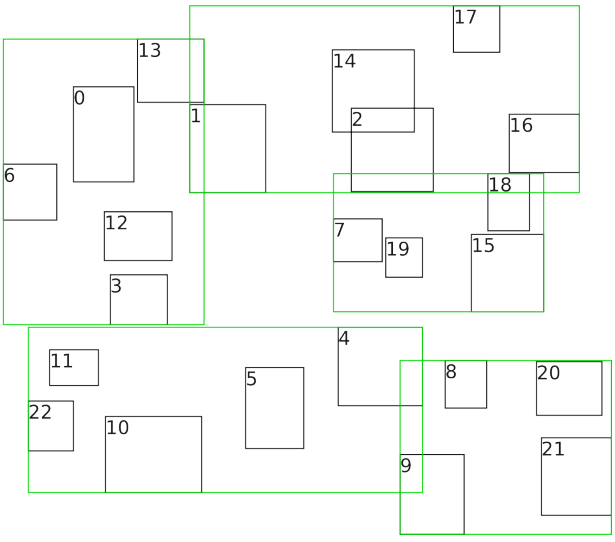


Figure 3: Screenshot from the application. The R-Tree used is a 2/3 (min records per node/max records per node). All the layers of bounding boxes are visible. With black color are depicted the shapes and with other colors the tree's bounding boxes of different heights.



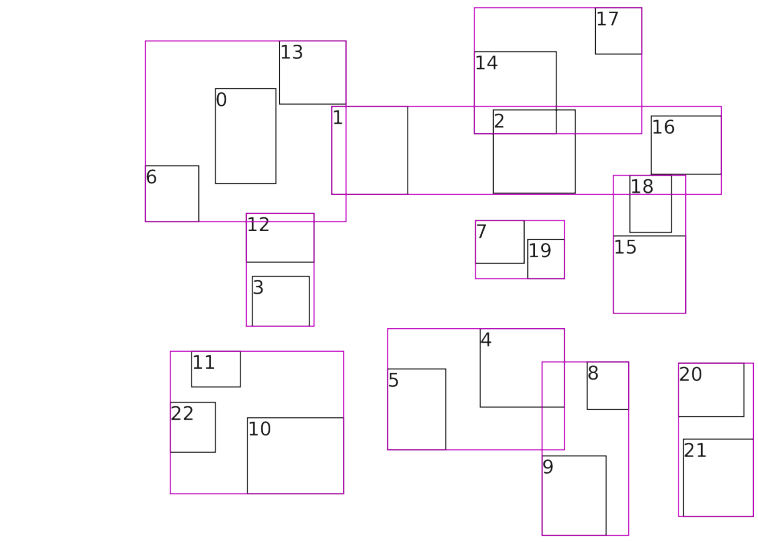
'I': Insert, 'U': Update, 'S': Search, 'D': Delete
'T': Bounding Boxes: ON, '0'-'8', 'P': Show Layer: 0
Mode: Insert (Click whitespace and Drag)

Figure 4: Same as Figure 3. Only layer 1 of bounding boxes is visible.



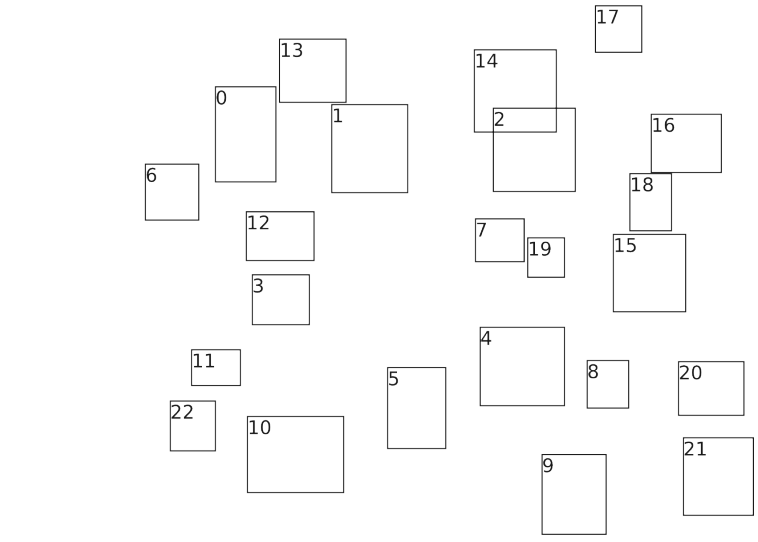
'I': Insert, 'U': Update, 'S': Search, 'D': Delete
'T': Bounding Boxes: ON, '0'-'8', 'P': Show Layer: 1
Mode: Insert (Click whitespace and Drag)

Figure 5: Same as Figure 3. Only layer 2 of bounding boxes is visible.



'I': Insert, 'U': Update, 'S': Search, 'D': Delete
'T': Bounding Boxes: ON, '0'-'8', 'P': Show Layer: 2
Mode: Insert (Click whitespace and Drag)

Figure 6: Same as Figure 3. Only layer 3 of bounding boxes is visible.



'I': Insert, 'U': Update, 'S': Search, 'D': Delete
'T': Bounding Boxes: OFF, '0'-'8', 'P': Show Layer: all
Mode: Insert (Click whitespace and Drag)

Figure 7: Same as Figure 3. The layers of bounding boxes are switched off.

3.2 Test

The `rtree-test` application uses lib2geom's R-Trees and performs some insert and erase operations. First, a number of rectangles are inserted (up to 30K) and then we erase randomly 10% of them. After each insert and erase operation we perform a sanity check to make sure that the tree is not malformed.

This application is also a demo for the usage of the lib2geom's R-Trees.

4 Deliverables

The source code of the project is licenced with the same licence as lib2geom, LGPL v2. The code is available with lib2geom at:

`http://sourceforge.net/projects/lib2geom`

The files related to the project are:

- `lib2geom/src/2geom/rtree.*`: rtree implementation.
- `lib2geom/src/2geom/toys/rtree-*`: GUI and test cases.
- `lib2geom/src/2geom/redblack*`: attempt to implement red-black trees and interval trees. Insertion works in both cases.

The code is fully commented and this report is an additional form of documentation.

5 Conclusion

The project's goals were successfully covered. The R-Tree was implemented, the test cases don't reveal any problems and we assert that the properties of the R-Tree are maintained throughout consecutive inserts and erases.

5.1 Future Work

The developers of Inkscape seem to be satisfied with the implementation, so we are now discussing how the lib2geom R-Trees can be further expanded and integrated in Inkscape. One of the features we would like to add is the ability to write and read efficiently from the disk. The author doesn't have experience on the subject, so any further steps to this direction need carefull planning.

An other interesting addition that was discussed with Nathan Hurst is the incorporation of mimaps in the R-Tree. During rendering, we could take into account the level of detail. When a shape is too small compared to the user's window, instead of rendering the actual vector shape, we could draw a pre-rendered image of the shape, thus saving rendering time.

The understanding of R-Trees and the challenges of spatial indexing is an two major assets gained throughout this project. Another highly satisfactory outcome is that this project to lib2geom, could improve one factor of Inkscape's performance, thus contributing to the community effort for a powerfull and user friendly open source 2D vector graphics editor.

References

[1] ELLAK. Contest results for implementing open source software. http://www.ellak.gr/index.php?option=com_openwiki&Itemid=103 &id=eellak:foss_diagonismoi_apotelesmata.

[2] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984*. ACM Press, 1984.

[3] Inkscape. Inkscape official web site. http://www.inkscape.org/.

[4] Libcairo. Libcairo official web site. https://launchpad.net/libcairo.

[5] Y. Manolopoulos, A. Nanopoulos, A.N. Papadopoulos, and Y. Theodoridis. *Rtrees: Theory and Applications*. Series in Advanced Information and Knowledge Processing. Springer, 2005.

[6] MySQL. Mysql 5.0 reference manual 11.12.6.1. creating spatial indexes.

[7] Oracle. Oracle spatial data sheet: Option location-based services for oracle9i.

[8] PostgreSQL. Postgresql 8.1.17 documentation 11.2. index types.

[9] Shashi Shekhar and Sanjay Chawla. *Spatial Databases: A Tour*. Prentice Hall, 2003.