3D SPATIAL INDEXING OF OBJECTS

Miroslav Drbal
Master Degree Programme (2), FIT BUT
E-mail: xdrbal03@stud.fit.vutbr.cz

Supervised by: Filip Orság
E-mail: orsag@fit.vutbr.cz

ABSTRACT

This paper introduces the reader to the issue of spatial indexing of objects, also discusses differences between indexing static and moving objects in the 3D space. The main part is dedicated to the MaNGOS's data analyze and sketches out the implementation of the 3D spatial index aimed to be used as replacement of the grid index[3] in open-source application MaNGOS[1].

1 INTRODUCTION

At this point I would like to outline what the reader can imagine beyond the term *spatial index*. The spatial index[2] is a data structure created upon the set of objects which makes some operations on a key attributes more faster. For example we can consider searching for all objects of the given properties in the certain subspace of the whole operating space[3]. With the non-indexed very simple approach we have to traverse the whole collection of objects in the entire space and check the conditions. This is very ineffective for large sets of data. And this is the very point where the indexing structures becomes very useful.

There is also a big difference in the indexing algorithms when the character of data is static or we have to deal with the moving objects. In fact every spatial index is based on dividing space into the smaller subspaces and organizing them into the linked structures (mainly into the tree structures). This indexes are very effective for searching operations upon the static data. They are based on an idea that the subspaces have constant size in time. If we try to use this index on the set of mainly moving objects we get very often updated index what leads to very often subspaces rebuilding and objects reinsertioning. This causes performance degradation of the whole index.

For more details I would like to redirect the reader to bibliography articles [3, 1, 2, 4] for more information about the indexing structures for static data and articles[5, 6] for moving data.

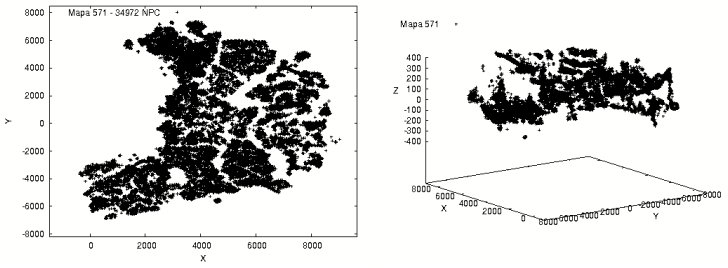
[1] Project can be tracked on its homepage at http://www.getmangos.com
[2] Not only spatial but theoretically any other index.
[3] Find all pizza restaurants in the radius of 1km from my current position

2 MANGOS DATA ANALYSIS

MaNGOS[4] is open-source application distributed under GNU/GPLv2 license. It is network server for the MMORPG[5] game World of Warcraft™.

The game environment consists of four base continents[6] and the smaller maps with dungeons. The game objects can be divided into a three main groups: *GameObjects*, *NPCs*, *Players*. Each continent map contains approximately 25000 *NPCs* and *GameObjects*.

Now let aim our attention to how this game objects are distributed in the map. The base size of each map is $34133,33 \times 34133,33$. If we consider the point $[0;0]$ as the center of the map we are getting the Cartesian coordinated system with bounds from interval $(-17066,665; 17066,665)$. The resolution of this system is given by the precision of *float* data type. On the picture 1 we can see how NPC objects are flat and spatially distributed on the map 571[7]. Each black cross



(a) Map 571 **(b)** Map 571 3D

Figure 1: *Flat and spatial NPCs distribution on map 571*

represents one NPC object in the space. The shape of the continent is not explicitly drawn in the picture, but can be really well guessed from the NPC density. The table 1 characterizes the data from the side of its static or moving character in the world scale. As we can see the most of the data can be considered as static.

Object Type	Moving [%]	Static [%]	Count in world
GameObject	0	100	136931
NPC	22	78	139037
Player	100	0	2000
Summary	11,7	88,3	277968

Table 1: Percentage summary of moving and static objects in world scale.

3 DESIGN OF INDEXING STRUCTURE

With regard to the data analysis I decided implement the indexing structure as a hybrid combination of the R*-Tree[4] for indexing static data and the kD-Tree[2] for indexing moving data. Each bounding box in the R*-Tree contains pointers to the kD-Tree subspaces where it belongs

[4]Massive Network Game Object Server
[5]Massive(ly)-Multiplayer Online Game
[6]Kalimdor, Eastern Kingdoms, Outland, Northrend
[7]This map number belongs to Northrend continent

to[8]. In the standard index application where we index a common data, there is no direct linking between the object and the index. In the MaNGOS I would like to introduce the backpoiters from the game objects to the R*-Tree's bounding boxes what can improve searching performance for the small radiuses because we don't need to traverse whole the structure from the root node and it can also improve performance for the middle radiuses where we can traverse the tree from bottom up.

Implementation will be done in the C++ with use of the template classes and the meta-programming. The leaf nodes, containing the pointers to the data, are designed as the universal data storage. Each stored data type will have its own container transparently accessible by the indexe's methods. This approach has few disadvantages like not trivial implementation of searching for multiple types at the same time. On the other hand there is no need to have a common predecessor for the all indexed objects, no need to implement the object type identification on the object level and at last no need of casting a common predecessor to a correct data type on the query return.

4 CONCLUSION

Further testing will be needed to prove the performance increase with this new indexing structure against the default grid index. Also some minor problems will need to be solved. For example the algorithm for classification currently active objects[9].

REFERENCES

[1] Apostolos N. and Papadopoulos,Yannis Manolopoulos. *Nearest neighbor search: a database perspective*. Springer Science+Bussines Media, Inc., 2005. ISBN 0-387-22963-9.

[2] Hemant M. Kakde. Range searching using kd tree. `www.cs.fsu.edu/ lifeifei/cis5930/kdtree.pdf`, 2005-08-25 [cit. 2010-01-01].

[3] Kevin Sahr, Denis White, and A. Jon Kimerling. Geodesic discrete global grid systems. `http://www.sou.edu/cs/sahr/dgg/pubs/gdggs03.pdf`, 2003 [cit. 2009-12-27].

[4] Wikipedia. R*-tree. `http://en.wikipedia.org/wiki/R*_tree`.

[5] Yufei Tao, Dimitris Papadias, and Jimeng Sun. The tpr*-tree: An optimized spatio-temporal access method for predictive queries. `www.cs.ust.hk/ dimitris/PAPERS/VLDB03-TPR.pdf`, [cit. 2009-01-04].

[6] Yuni Xia and Sunil Prabhakar. Q+rtree: Efficient indexing for moving object databases [online]. `www.cs.purdue.edu/homes/sunil/pub/yuniDASFAA03.pdf`, [cit. 2009-01-03].

[8] I expect the size of the R*-Tree's bounding box will be $<<$ to the kD-Tree's subspace so there can be maximally 4 pointers in the worst case. Also bounding box creation heuristics can minimize this pointers count to 1

[9] Currently it is done by flagging all objects in all grids around the grid with the player as active.