Data Structures for geolocalization of robotic bees and avoiding crashes between them

María Camila Morales Ríos
EAFIT University
Colombia
mcmoralesr@eafit.edu.co

Mauricio Toro
EAFIT University
Colombia
mtorobe@eafit.edu.co

ABSTRACT:

The target of this paper is to show a solution for a futuristic related problem that is a concerning one for humankind: the disappearance of the Anthophila species. With that on mind, the main aspect that we describe here is the possibility of creating robotic bees that will held the work of doing pollination.

With the growth of mass extinction of bees and this solution being one alternative for rebuilding what they are doing, it will be a challenge to localize each individual bee and avoid crashes between all of them, for example, if we talk of a large amount of these Robots working at the same time in a particular area with Geodetic coordinates being received, for that reason, the principal obstacle will be solutioned creating a data structure to detect possible collisions between bees that are less than 328.084 feet away from each other, some examples of algorithms and data structures that solve this problem are mainly used in the videogame industry and I will describe them in the following development of this research.

1. INTRODUCTION:

Based on the dramatic drop of bee population that started in 1997 and got worse during the year 2005, the constant use of pesticide and some parasites were slowly eradicating the specie; an issue had begun for agricultural workers who depended on bees to pollinate crops and, during that era and the following years, importing bees was a provisional solution for them. [1]

Recently, some Universities had been interested on using Robotic Pollinators to stop the problem of bees being exterminated and not being capable of find another specie capable of doing the work that they did, for that, a general question is ¿how the robotic pollinators will be located and avoid collisions between them? For that, the point of this whole investigation is to find a way to solve the inquiry.

2. PROBLEM:

The difficulty of this problem is to find a precise data structure for the robotic bees that will not allow wrecks among them, we are solving this problem because is not a distant problem that computer science will investigate in the future, is a daily matter of interest to put effort into, and is a problem often used in other areas, is an adequate example of issues that will appear on other fields that need constant alternatives that are both efficient and effective.

3. RELATED JOBS

3.1 Quadtree

A quadtree is a tree data structure in which each internal node has exactly four children. Quadtrees are the two-dimensional analog of octrees and are most often used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions. The data associated with a leaf cell varies by application, but the leaf cell represents a "unit of interesting spatial information". [2]

- A quadtree starts as a single node. Objects added to the quadtree are added to the single node.

- When more objects are added to the quadtree, it will eventually split into four subnodes. Each object will then be put into one of these subnodes according to where it lies in the 2D space. Any object that cannot fully fit inside a node's boundary will be placed in the parent node.

- Each subnode can continue subdividing as more objects are added.[3]

3.2 Spatial Hashing

A spatial hash is a 2 or 3 dimensional extension of the hash table. The basic idea of a hash table is that you take a piece of data (the 'key'), run it through some function (the 'hash function') to produce a new value (the 'hash'), and then use the hash as an index into a set of slots ('buckets').

To store an object in a hash table, you run the key through the hash function, and store the object in the bucket referenced by the hash. To find an object, you run the key through the hash function, and look in the bucket referenced by the hash.

Typically, the keys to a hash table would be strings, but in a spatial hash we use 2 or 3 dimensional points as the keys. In addition, here is where the twist comes in: for a normal hash table, a good hash function distributes keys as evenly as possible across the available buckets, in an effort to keep lookup time short. The result of this is that keys which are very close (lexicographically speaking) to each other, are likely to end up in distant buckets. However, in a spatial hash we are dealing with locations in space.[4]

3.2 AABB Trees

An AABB tree is nothing but simply a binary tree, where all the AABBs are stored at the leaves. The main advantage for this kind of broad-phase is that this is a border-less data structure, and it doesn’t require you to explicitly specify an area which other kinds of data structures such as grids or QuadTrees require.

A Dynamic AABB Tree is a binary search tree for spatial partitioning. [5]

It is very efficient in all aspects in terms of game physics, including ray casting, point picking, region query, and, most importantly, generating a list of collider pairs that are potentially colliding (which is the main purpose of having a broadphase) Each collider has its own AABB, and they are stored inside a binary tree as leaf nodes. Each internal node, a.k.a. branch node, holds its own AABB data that represents the union of the AABB of both of its children. [6]

3.4 R-Tree

R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons.

The key idea of the data structure is to group nearby objects and represent them with their minimum bounding rectangle in the next higher level of the tree; the "R" in R-tree is for rectangle. Since all objects lie within this bounding rectangle, a query that does not intersect the bounding rectangle also cannot intersect any of the contained objects. At the leaf level, each rectangle describes a single object; at higher levels the aggregation of an increasing number of objects. This can also be seen as an increasingly coarse approximation of the data set.

Similar to the B-tree, the R-tree is also a balanced search tree (so all leaf nodes are at the same height), organizes the data in pages, and is designed for storage on disk (as used in databases). Each page can contain a maximum number of entries, often denoted as {\displaystyle M} M. It also guarantees a minimum fill (except for the root node), however best performance has been experienced with a minimum fill of 30%–40% of the maximum number of entries (B-trees guarantee 50% page fill, and B*-trees even 66%). The reason for this is the more complex balancing required for spatial data as opposed to linear data stored in B-trees.[7]

-Every leaf node contains between m and M index records unless it is the root

- For each index record in a leaf node, I 1s the smallest rectangle that spatially contains the n-dimensional data object represented by the indicated tuple

- Every non-leaf node has between m and M children unless it is the root

- For each entry in a non-leaf node, I 1s the smallest rectangle that spatially contains the rectangles in the child node

- The root node has at least two children unless it is a leaf

- All leaves appear on the same level [8]

REFERENCES

1. Bees in peril: a timeline, Margaret Badore, Retrieved February 24, 2018 from https://www.treehugger.com/sustainable-agriculture/bees-peril-timeline.htm

2. Quadtree, Wikipedia, Retrieved February 24, 2018 from https://en.wikipedia.org/wiki/Quadtree

3. Quadtrees to Detect Likely Collisions in 2D, Retrieved February 24, 2018 from Space.https://gamedevelopment.tutsplus.com/tutorials/quick-tip-use-quadtrees-to-detect-likely-collisions-in-2d-space--gamedev-374

4. Spatial Hashing, Retrieved February 24, 2018 from https://www.gamedev.net/articles/programming/general-and-gameplay-programming/spatial-hashing-r2697/

5. AABB Trees for Collision Detection, Retrieved February 24, 2018 from https://goharsha.com/blog/aabb-trees-for-collision-detection/

6. Game Physics: Broadphase – Dynamic AABB Tree, Retrieved February 24, 2018 from http://allenchou.net/2014/02/game-physics-broadphase-dynamic-aabb-tree/

7. R-tree, Wikipedia, Retrieved February 24, 2018 from https://en.wikipedia.org/wiki/R-tree

8. R-trees: a dynamic index structure for spatial searching Retrieved February 24, 2018 from, https://dl.acm.org/citation.cfm?id=602266