Collisions between robotic bees: a data structure on how detect them efficiently.

Carla Daniela Rendon
Universidad Eafit
Colombia
cdrendonb@eafit.edu.co

Sebastian Arboleda Botero
Universidad Eafit
Colombia
sarboledab@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

SUMMARY
In this project we want to implement a data structure that efficiently detects collisions between robotic bees, in order to find a solution to this problem we must take into account certain factors (such as execution time and efficiency) since this algorithm must work effective way in a daily environment. Solving this problem will help prevent possible damages and losses. This type of problems is often seen in video games when it is identified that there is a collision of one object or character against another.

1. INTRODUCTION

For no one is a secret, that the human being, making use of his ignorance and lack of awareness, has caused irremediable damage to the environment. For several years now there has been a problem, of multiple causes (uncontrolled use of pesticides over the last decades, parasites and deforestation) and with extremely devastating consequences, ranging from the disappearance of food to the destruction of the balance that today it exists in nature, we refer to "The extinction of bees", a theme that has been the subject of intense debates on the part of scientists. Different solutions have been proposed to this problem, and beyond the awareness campaigns, we talk about the creation of robotic bees, which can fulfill the same functions of a normal bee, however, to create these small robots, they must take into account different factors, and beyond that they are able to pollinate the plants, they must also have the ability to coordinate with each other to avoid possible collisions and losses.

2. PROBLEM

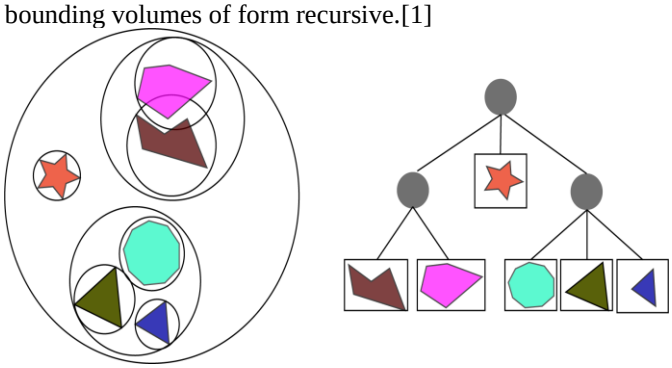
The problem to be solved is to detect the collision between robotic bees by creating a data structure. It is very important to solve this problem, since, otherwise, multiple losses would occur and the work could not be completed efficiently.

3. RELATED JOBS

Works or studies related to the collision of objects, which could be used for the solution of our work.

3.1 bounding volume hierarchy (BVH)

The BVH deals with the solution to a problem of hierarchy of volumes, used for the detection of ray tracing collisions which is a way of generating an image by tracing the path of light as pixels in an image plane and simulating the effects of his encounters with virtual objects. The solution that the BVH gives us is to provide the bounding volume as leaves of the tree that are then grouped as small sets that are included within larger bounding volumes and said volumes, they are also grouped and enclosed within other larger bounding volumes of form recursive.[1]

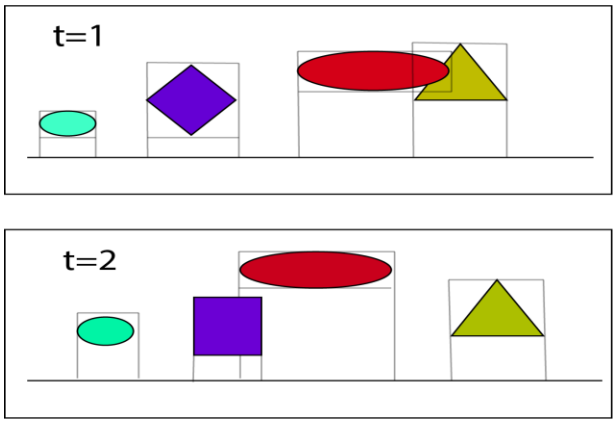


Example taken From [1]

3.2 AABB Tree Collision Detection

Tree AABB or Tree of Bounded Boxes Aligned by the Axes, it is a data structure that seeks to perform intersections and calculate distances between geometric objects in 3D with a finite number of points. Its implementation is given by building an initial set of elements (points, triangles, polygons, etc.). with this the set is ordered along the largest of the coordinate axes and separating into two equal or balanced subsets, applying this method recursively until reaching the base case, and several of these base cases are usually taken. [2]

Example taken from [2]



3.3 Método de Sean Quinlan

It consists of using envelope representations based on spheres, these spheres can be specified with a position vector and a radius, the calculation of the distance between two spheres does not entail many inconveniences or expenses.

Its hierarchical structure consists of a binary tree approximation, where each node contains a sphere, the tree has two properties:

• The union of all spheres completely contains the surface of all objects.

• The sphere of each node contains the spheres of its descendant nodes.

A collision detection algorithm plays a very important role within a simulation system, because for each movement that the objects make, it must be verified if there is a collision or not. The method of Sean Quinlan proposes to solve this problem quickly and economically. [3]

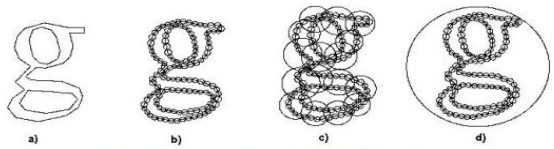


figura 2.1 Algunos niveles de detalle del árbol de esferas[1]

Example taken from [3]

3.4 Método de Philip M. Hubbard

It is a method to approximate objects that support collision detection algorithms in critical time. The approximations are hierarchical spheres, which allow a critical time algorithm to progressively refine the accuracy of its detection, stopping when necessary to maintain the essential functioning of interactive applications in real time. The key to critical time collision detection is based on a method that automatically approximates the surface of an object, this is based on constructing the hierarchical structures that model the surface of objects during a preprocessing stage. The solution proposed by this algorithm is more than anything else, to refine the work done by the collision detection algorithms. [4]

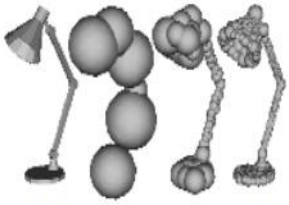


Figura 2.2 Una lámpara con 626 Triángulos y tres niveles de detalle usando esferas [2]

Example taken from [4]

4. AUTHOR KEYWORDS:

Algorithm: is a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

Data structure: in computer science, a **data structure** is a particular way of organizing and storing **data** in a computer so that it can be accessed and modified efficiently.

ArrayList: The ArrayList class extends AbstractList and implements the List interface.ArrayList supports dynamic arrays that can grow as needed.

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.

5. ACM KEYWORDS.

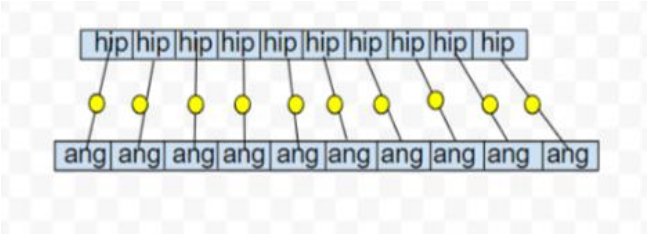
1. Information systems---Database management system engines

2. Software and its engineering---Software notations and tools---General programming languages---Language features---Inheritance

3. Software and its engineering---Software notations and tools---General programming languages---Language features---Classes and objects

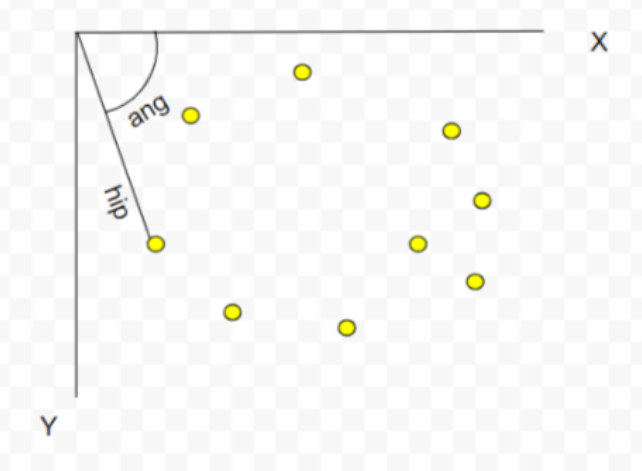
6. DESIGN OF THE DATA STRUCTURE. (SOLUTION 1).

Algorithm: https://github.com/carlashawol/ST0245-032/blob/master/proyecto/codigo/main.cpp



For the implementation of the first solution to the problem: "Detect collisions between robotic bees", we decided to make a data structure that counts two arrays, in the first the hypotenuse is stored and in the second the angle that produces the point (bee) with respect to the point (0,0). In this algorithm the polar coordinates are used to detect the position of the bee in the 2D plane, once the bees are inserted randomly, the positions are stored in the two arrangements and they are bought with the new bees that enter to the plane, if one of the new positions coincides with an earlier one already stored, it is concluded that the bees collided.

7. DESIGN OF THE OPERATIONS OF THE DATA STRUCTURE. (SOLUTION 1)



As we can observe in the graph previously shown, the point of origin (0,0) of the Cartesian axis, is positioned in the upper left part of the screen, each bee has an angle and a hypotenuse with respect to the point of origin, which are calculated in order to obtain the polar coordinate of the bee, the hypotenuse and the angle formed by each bee with respect to the point (0,0) are stored in two arrays (in one the hypotenuse and in another the angle) and the previous positions are bought with the new positions of the bees that are inserted randomly in the plane, if a previously stored polar coordinate is equal to the new polar coordinate that is being stored, then it can be concluded that those two bees They have collided.

8. COMPLEXITY CALCULATION

Operations	Complexity
Insertion of the angle of the bees	O(n)
Insertion of the Hypotenuse of bees	O(n)
Comparison and collision detection	O(n^2)

9. DATA STRUCTURE DESIGN CRITERIA.

We decided to implement this data structure as the first solution, because, due to its simplicity, ease of implementation and complexity, it provided us with a base with which to build the final algorithm. As main advantages of this algorithm we can highlight its objectivity when it comes to demonstrating the results and ease of checking these, as the main disadvantage we have the complexity O (n2), the slowness that the program presents when executing operations is a great inconvenience , especially if you want to perform an algorithm with inputs similar to reality. As mentioned at the beginning of the project, certain factors must be taken into account for the realization of the data structure since it must function efficiently in a daily environment.

10. RESULTS OBTAINED.

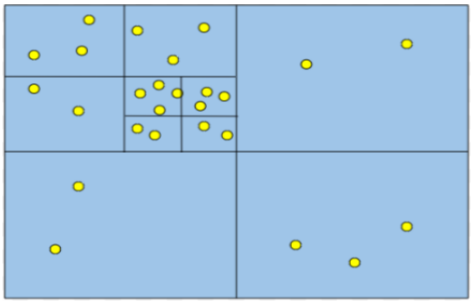
Self- Completed Structures	Insertion of all bees	Detection of the first collision
Execution time. (100 bees)	17 s	2s
Execution time. (1000 bees)	45s	2s
Execution time.. (10000 bees)	128s	3s
Execution time. (100000 bees)	337s	2s
Execution time.. (1000000 bees)	488s	3s

From the results obtained we can conclude that the insertion of all the bees in the plane is a delayed process, and that it increases proportionally with respect to the number of bees that we wish to insert, on the other hand the detection of the first collision is not seen very affected by the increase in the number of bees that you want to insert, this because the insertion is random and the first overlap of one bee with another, is not directly related to the amount of bees that you want to insert.

11. PROBLEM SOLUTION (SOLUTION 1)

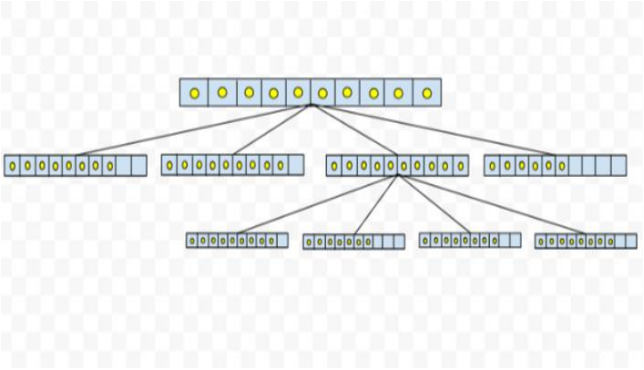
Algorithm: https://github.com/carlashawol/ST0245-032/blob/master/proyecto/codigo/main.cpp

12. DESIGN OF THE DATA STRUCTURE (FINAL SOLUTION).



As we can see in the graph, for the final solution to the problem we decided to implement the algorithm known as Quadtree, this consists of dividing the plane (root) into 4 parts (child nodes) when a considerable number of bees is accumulated in the space to be taken consider. The Quadtree is an algorithm that is frequently used in video games to detect collisions between objects and characters, this is because it is easy to implement, and its complexity is O (log4 (n)).

13.DESIGN OF THE OPERATIONS OF THE DATA STRUCTURE. (FINAL SOLUTION)



For the realization of this algorithm we consider the complexity of this, since when carrying out a program with inputs like reality (10,000,000 of bees, for example), it is necessary that the operations are carried out quickly and effectively, the algorithm known as Quadtree offers a complexity O (log4 (n)), which allows great speed when implementing operations within the program. On the other hand, we decided to also implement an ArrayList within each node of the tree, to clearly define when a new division in the plane must be made, in our case each ArrayList has a maximum of 10 spaces, when an eleventh bee tries to enter the node, this is divided into 4 new ArrayList (nodes).

14. COMPLEXITY CALCULATION. (FINAL SOLUTION)

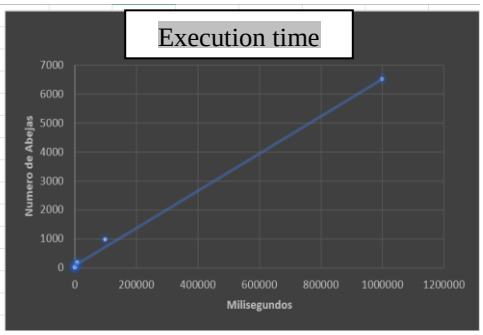
Operations	Complexity.
Lector	O(n)
Transform Strings into files	O(n/2)
Quadtree	O(Log4(n)
ArrayList	O(1)
Comparison of bees	O(10^2)

15. DATA STRUCTURE DESIGN CRITERIA. (FINAL SOLUTION).

As we mentioned in number 13, the Quadtree is an algorithm that has a complexity O (log4 (n)), when you want to make a program with inputs similar to reality, it is necessary to take into account the complexity of the algorithm, if it has a very high complexity, as it was the case of the first implemented solution, which had a complexity O (n ^ 2), if we observe in the results obtained from number 10, the process of inserting the bees, when they are taken into account Similar entries to the reality (1000000 of bees, for example), is affected and the execution time increases exponentially. On the other hand, we decided to implement an ArrayList in each of the nodes of the tree, with a maximum of 10 spaces (bees), if this number of bees is exceeded in that node, it is divided into four parts. When a maximum of 10 spaces is defined for each arrangement, when comparing them, the complexity of the algorithm is not affected, since it goes from O (n ^ 2) or O (10 ^ 2).

16. RESULTS OBTAINED. (FINAL SOLUTION)

10	6,462048
10	6,462048
100	9,919091
1000	19,82767
10000	165,345268
100000	955,2793479
1000000	6.493



We can conclude from the graph previously shown, that there is an exponential increase in the execution time, but this is less than the first solution.

17. CONCLUSIONS

The disappearance of bees is a problem that has been presented for several years. We are in the era of technology and many of the problems that afflict human beings are solved by technological resources. The creation of robotic bees has been proposed as a possible solution, in this project we designed an algorithm to detect collisions between them.

Two solutions were implemented, in the first, the results obtained were not very favorable, with a very high execution time and complexity. For the final solution, we managed to reduce the complexity and the execution time with the application of a Quadtree, going from a complexity O (n ^ 2) to O (log4 (n)).

At the beginning of the project, we mentioned 4 possible algorithms to be applied to solve the problem, none of these algorithms was used for the final solution of the problem, since its implementation was very complicated, the Quadtree is presented as a simple solution to implement and with a favorable complexity.

In this project an algorithm was proposed to detect collisions between robotic bees, as a continuation and future complement for this algorithm could be designed a data structure that not only detects but avoid collisions between robotic bees.

18. THANKS TO:

This research was supported supported by R.C.J. Services. S.A.S.

19. REFERENCES

1.Bounding volume hierarchy (BVH), from https://en.wikipedia.org/wiki/Bounding_volume_hierarchy#Usage

2.Simulación de colisiones con deformación en paralelo, Victor León Higuita Cardona y Juan Diego Toro Cano, year 2010, Universidad EAFIT, pag.35.

3. AABB Tree Collision Detection, taken from, https://www.azurefromthetrenches.com/introductory-guide-to-aabb-tree-collision-detection/

4. Método de Sean Quinlan y Philip M. Hubbard, taken from:http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/ramirez_o_md/capitulo2.pdf

20. TEAMWORK.

Since the beginning of the semester, the work was distributed equally, Sebastián Arboleda, researched, for the first 2 solutions, the bounding volume hierarchy (BVH) and the AABB Tree Collision Detection, meanwhile, Carla Rendón, investigated the Method of Philip M. Hubbard and the Sean Quinlan Method. The introduction, the summary and the approach of the problem, was carried out by both team members. Carla Rendón was responsible for making the numeral 4 (author keywords) and Sebatian arboleda the numeral 5. The sections of the first solution were implemented (6,7,8,9,10,11) by Carla Rendón. The sections of the final solution (12,13,14,15,16) were implemented by Sebastián Arboleda. The conclusions and final notes were made by both team members.

21.FINAL PROBLEM SOLUTION.

Algorithm: https://github.com/carlashawol/ST0245-032/blob/master/proyecto/codigo/ProyectoFinal.rar