POLLINATORS DRONES: A COLLISION DETECTION ALGORITHM

Daniel Hincapié
Universidad Eafit
Colombia
dahincapis@eafit.edu.co

Anthony García
Universidad Eafit
Colombia
agarciam@eafit.edu.co

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

ABSTRACT

Bees are too important for us, if they disappear, we will do it with them. Therefore, it is necessary to implement drones that are responsible for carrying out the task that this species executes, pollination. This is where our potential as engineers becomes visible and we see ourselves in the responsibility of creating an algorithm that prevents these drones from colliding with each other. For this we have adapted the octree, a data structure that allows us to know the position of the drones to avoid the collision of these. The octrees are really useful, because they offer us enough efficiency when working with large proportions of data, making their implementation successful.

Keywords

Algorithm, Data structures, Octree, bees, Linked Lists, Collision.

ACM CLASSIFICATION Keywords

Example: Theory of computation → Design and analysis of algorithms → Data structures and algorithms → Collision detection

1. INTRODUCTION

The bees are responsible for pollinating a vast variety of crops, providing a good number of species, including humans, food.

The vital importance of bees becomes evident. However, in recent years, there has been a huge reduction in its population, becoming a worrying topic. Nothing else in the United States, there was a reduction of 40% of bees, in 2012.

The cause of this decrease is linked to parasites, the constant use of pesticides, global warming, deforestation, among others.

The human species is at risk from the disappearance of bees, so a motivation arises to find a solution to this growing phenomenon.

2. PROBLEM

The constant reduction of bees, would generate, with the lack of pollination, an shortage of food that would eventually lead to the end of the human being and many other species.

The social impact would be terribly high, therefore it is necessary to look for alternatives to solve this global problem. Undoubtedly, one of the best, is the use of drones that pollinate, however, these machines could collide, raising the problem of how to prevent this from happening.

3. RELATED WORK

3.1 Collision detection in 2D video games

In video games, it is necessary to know when one object collides with another, however, many algorithms tend to slow down the video game and make it less than optimal.

For this the quadtree arise, a structure of data, that by means of the division of the elements in quadrants, provides optimal results at the time of detecting collisions.

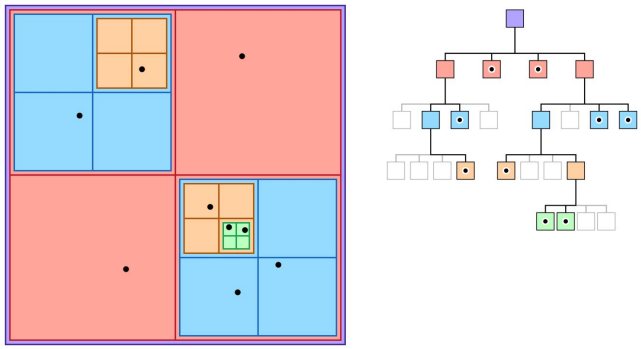


Figure 1: Representation of a quadtree.

3.2 Collision detection in 3D simulation

In simulations and 3D objects, it is important to detect the simulations, however, not all data structures work properly for 3D objects.

In this case, we can make use of the Octrees, data structures that allow us to find the collisions quite easily and without abusing the time and memory of our machine, offering us a lot of optimization.

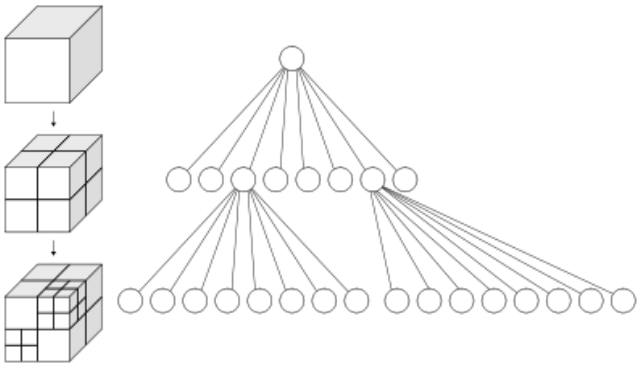


Figure 2: Representation of an octree.

3.3 Hash tables in cryptography

When we work, for example, with bitcoins, it is possible to have collisions. This happens when two input values have the same summary. We can solve this easily by applying again a data structure to our algorithm. Hash tables resist these collisions and solve our problem.

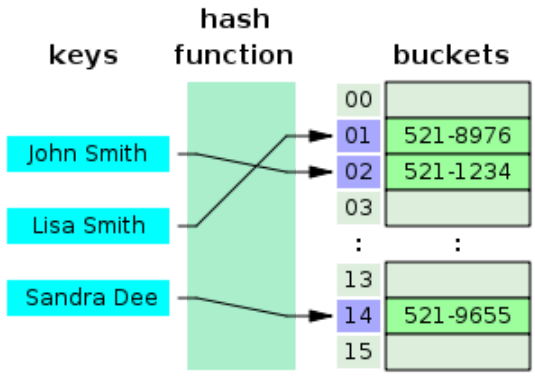


Figure 2: Representation of a hash table.

3.4 Automated drones collisions

When working with automated drones, there is a risk that they could collide. However, there is an algorithm to avoid these collisions between drones. UTM (Unmanned aircraft

4. Data structure: Octree and LinkedList

The following images explain the operation of the octree that we have implemented:

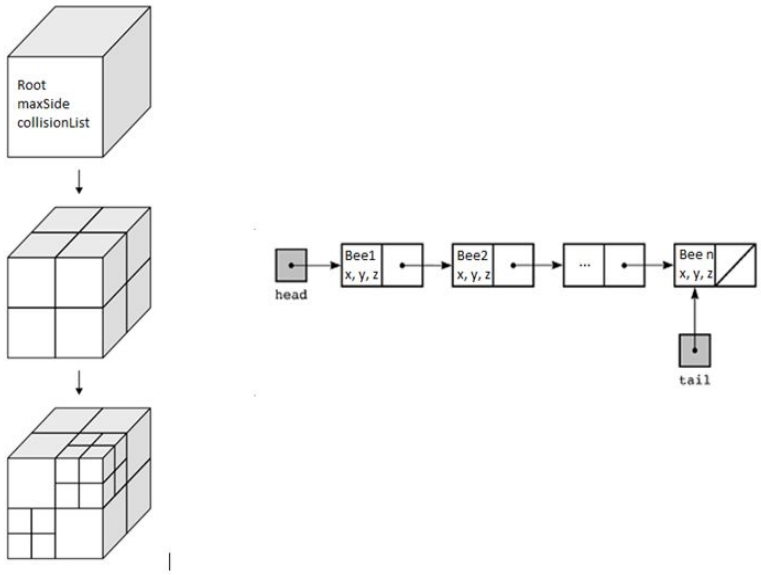


Figure 1: LinkedList with bees and the octree, the selected data structure.

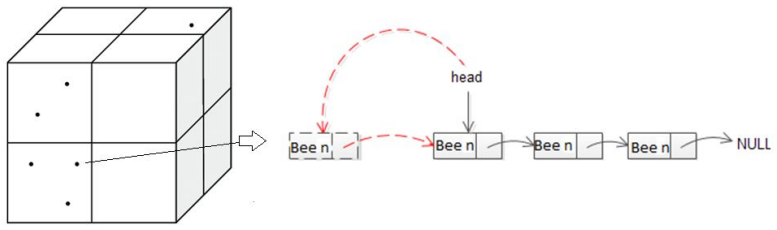
4.1 Operations of the data structure

Read file:

This operation is responsible for reading a file with different coordinates of bees, to later detect the collisions.

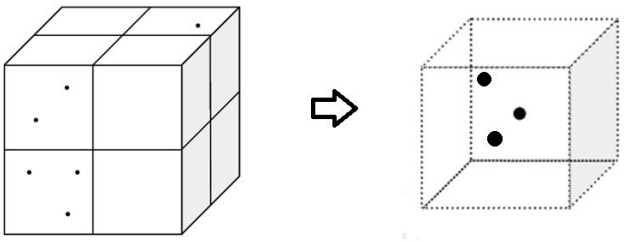
Insert:

This operation insert the bees and its coordinates in the Octree.



Detect collisions:

Analyze all the coordinates we have, in search of collisions.



Write files:

Writes a new file with the coordinates of bees at risk of colliding.

4.2 Design criteria of the data structure

The octree are really efficient data structures, and this is verified with the first implementation we did of this, however, in the previous case, when implementing it with an arrayList, the complexity, and therefore, the execution time and the spent memory, increase. In this case, when implemented with a LinkedList, its use is much more effective, leaving much better results.

4.3 Complexity analysis

Operation	Complexity
Read File	O(n)
Insert	O(log(s))
Detect collisions	O(n*s)
Write File	O(n)

Table 5: Table to report complexity analysis

4.4 Execution time

Bello

Operation	100 bees	1000 bees	100000 bees	1000000 bees
Read File	6 ms	6 ms	72 ms	571 ms
Insert	3 ms	2 ms	41 ms	368 ms
Detect Collisions	0 ms	0 ms	1 ms	1 ms
Write	5 ms	3 ms	117 ms	954 ms

Table 6: Execution time of the operations of the data structure for each data set in Bello.

Colombia

Operation	150 bees	1500	150000	1500000
Read File	4 ms	11 ms	186 ms	1844 ms
Insert	0 ms	10 ms	483 ms	3950 ms
Detect Collisions	0 ms	1 ms	77 ms	636 ms
Write	4 ms	7 ms	5 ms	12 ms

Table 7: Execution time of the operations of the data structure for each data set in Colombia.

4.5 Memory used

	100 bees	1000 bees	100000 bees	1000000 bees
Consumo de memoria	1,63 MB	3 MB	24,25 MB	166,5 MB
	150 bees	1500 bees	150000 bees	1500000 bees
	2 MB	4,4 MB	188,8 MB	1436 MB

Table 8: Memory used for each operation of the data structure and for each data set data sets.

4.6 Result analysis

Seeing the results, we can realize the variable s (space) is very important, and affects a lot in the execution time and spent memory, because the tree has to subdivide himself more times, and by that the insertion and detect collisions methods suffer some time changes. And the number of bees are too very important, as important as space, because the number of bees determinate the algorithm base.

Also using LinkedList instead of ArrayList gives a plus to the code, affecting the complexity of all the methods and thus the time of execution and spent memory, mainly in higher values.

Its important to consider the variable space is denoted by the length of the biggest side dimension raised to the cube.

5. CONCLUSIONS

From this project, we can visualize the importance of the correct use of the algorithms to obtain optimal results, which can be applied in solving large and complex problems, and the need to correctly use the data structures, as the case requires. .

We experimented with different data structures, however, we considered that the last one to be applied, gave us the best results, was quick, with a complexity that is not very high and allows us to work with large amounts of data without any problem.

If we analyze our first solution, and relate it to the last one, we observe that the efficiency increases remarkably. At first, some things complicated the obtaining of effective results, as for example the use of arraylists, whose complexity, was not the most appropriate for this problem.

Despite the satisfaction present in the work carried out so far, we do not rule out future improvements in our code, which allow us to reach even more optimal results.

5.1 Future work

Currently, we consider that our work is well implemented and offers good results, however, we consider it appropriate to intensify our knowledge in data structures, in a way that allows us to find a much more effective way of solving the problem, with much faster and optimal results.

ACKNOWLEDGEMENTS

We thank for assistance with support in the decision to select an effective data structure to Kevin Herrera and Sebastián Arboleda for comments that greatly improved the manuscript.

REFERENCES

1. Dccia.ua.es. (2019). [online] Available at: http://www.dccia.ua.es/dccia/inf/asignaturas/RG/2002/trabajos/colisiones.pdf [Accessed 14 May 2019].

2. OroyFinanzas.com. (2019). *¿Qué es una colisión en criptografía? - Criptografía Bitcoin*. [online] Available at: https://www.oroyfinanzas.com/2015/02/que-es-colision-criptografia-bitcoin/ [Accessed 14 May 2019].

3. europapress.es. (2019). *Un algoritmo regulará el vuelo de drones sin controladores humanos*. [online] Available at: https://www.europapress.es/ciencia/laboratorio/noticia-algoritmo-regulara-vuelo-drones-controladores-humanos-20151211140634.html [Accessed 14 May 2019