ALGORÍTMO PARA EL MONITOREO DE LA DISTRIBUCIÓN DE LAS ABEJAS ROBÓTICAS

Isabel Urrego
Universidad EAFIT
Colombia
iurregog@eafit.edu.co

Juliana Lalinde
Universidad EAFIT
Colombia
jlalindev@eafit.edu.co

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

RESUMEN
Las abejas son fundamentales para el desarrollo correcto de un ecosistema. La disminución de su población es un fenómeno preocupante de la actualidad, y ha brindado protagonismo a la idea del desarrollo de abejas robóticas, que cumplan sus funciones. Uno de los factores claves para poder ejecutar este proyecto es el monitoreo constante y efectivo con los robots, de manera que su distribución sea la más óptima.

Esta distribución determina el beneficio a obtener, pero depende de una ejecución adecuada: no se pueden presentar colisiones entre las abejas que se desplacen por espacios cercanos (en un plano tridimensional). La necesidad de ser alertados del riesgo de colisión de las abejas robóticas es el motivo del desarrollo de un algoritmo que analice las posibilidades de colisión. Como caso base de desarrollo se tomó la ciudad de Bello, donde las abejas robóticas tienen riesgo de colisión si están a menos de cien metros.

1. INTRODUCCIÓN

Las abejas han llegado a ser catalogadas por muchos científicos como unos de los animales más importantes para los ecosistemas. Ellas son polinizadoras de muchas plantas, lo que contribuye a la reproducción de diversas especies y producción de alimento necesario para los seres vivos. En algunos casos incluso la reproducción de estas especies depende únicamente de las abejas.

Actualmente, diferentes factores han conducido a la disminución de la población de abejas, poniendo en riesgo a muchas especies y por lo tanto a ciertos ecosistemas. Esto ha alertado a los especialistas y ha dado lugar al planteamiento de algunas posibles soluciones a este enorme riesgo. Por ejemplo, la invención y uso de abejas robóticas.

Estas realizarían las mismas funciones de las abejas reales, siendo una ayuda para que las diferentes plantas puedan seguir reproduciéndose con regularidad y en grandes cantidades. Sin embargo, lograr esto implica que las abejas deben ser debidamente programadas y evaluadas para poder ser puestas en acción y para esto se deben tomar diferentes aspectos.

Uno de los principales factores que se debe tener en cuenta es el riesgo de colisión que existe entre las abejas, además de que para optimizar su funcionalidad las abejas se deben encontrar distantes entre ellas. [1]

2. PROBLEMA

El objetivo de este proyecto es plantear un algoritmo eficiente en cuanto al tiempo que permita ayudar el desarrollo de dichas abejas robóticas. En este caso solo se tendrá en cuenta un aspecto con el cual se deben programar estos pequeños robots, el riesgo de colisión.

El objetivo principal será desarrollar un programa que permita detectar y notificar cuando unas abejas se encuentran a una distancia menor o igual a 100 metros. La información que se tendrá acerca de las abejas son sus coordenadas en un espacio Euclídeo (de 3 dimensiones).

3. TRABAJOS RELACIONADOS

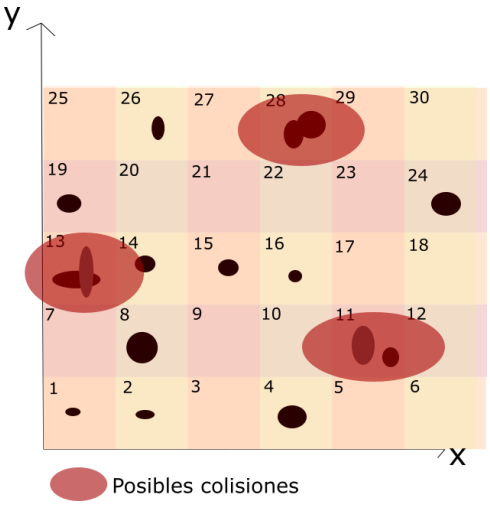
3.1 Spatial Hashing

Este algoritmo se utiliza para detectar colisiones entre objetos en un plano de dos o tres dimensiones con la ayuda de una estructura de datos conocida como tabla hash. Teniendo un conjunto de elementos, las tablas hash se encargan de asociar claves propias de cada elemento que se tiene con un valor. Además, tienen una función hash que se encarga de devolver un índice para poder acceder a un elemento de la tabla. [2]

Cuando dos elementos están guardados en la misma posición de la tabla se habla de una colusión y es precisamente este caso el que asocia las tablas hash con el problema de las abejas.

En el caso del hashing espacial, la tabla funciona como una especie de subdivisión del espacio y que haya una colisión significa que hay por lo menos dos elementos en la misma subdivisión.

Este algoritmo evita que se emplee una solución de fuerza bruta teniendo que comparar las distancias entre cada par de elementos del conjunto, se comparan únicamente los elementos de la tabla que contengan más de un elemento, pues la cantidad de elementos en una misma ubicación es lo que advierte sobre la colisión. De esta forma el algoritmo garantiza mayor eficiencia en cuanto al tiempo. [3]

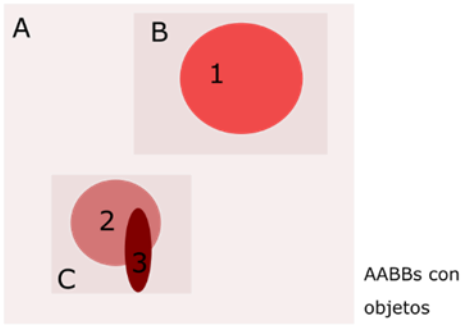


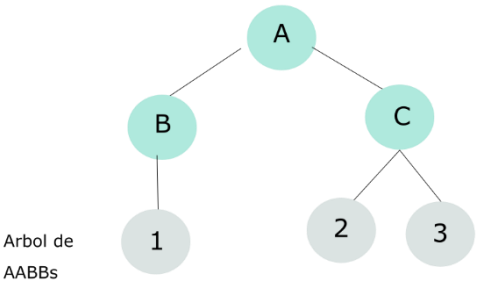
3.2 Árbol de AABBs

Este método se utiliza para verificar si hay colisiones entre diferentes objetos, con ayuda de AABBs.

AABBs son básicamente cajas que contienen objetos y de esta manera los ejes entre diferentes objetos se alinean. Es decir, en vez de tomar un objeto con todas sus regularidades, se toma el objeto como algo que está contenido entre ciertas coordenadas en x, ciertas en y y otro rango en z si se habla de tres dimensiones. Esto es para disminuir notoriamente el número de comparaciones que se deben hacer para identificar una colisión, pues usando AABBs solo se tendría que tener en cuenta un rango en cada eje. [4]

No obstante, comparar cada AABB cuando son numerosos elementos podría resultar muy poco efectivo y es por esto que se acude a los árboles. Estas estructuras de datos se forman asociando elementos (hojas) y juntándolos en una misma raíz que en este caso sería también un AABB. La ventaja está en el momento de revisar las3 posibles colisiones, pues simplemente se miraría si un objeto tiene riesgo con una raíz y solamente si la respuesta es sí, se procede a revisar cada "hoja" de la raíz para detectar dónde están las colisiones. Finalmente, no se estarían comparando todos los AABBs entre ellos, sino haciendo un número más reducido de verificaciones basadas en el árbol que se armó por medio de asociaciones de AABBs. [5]

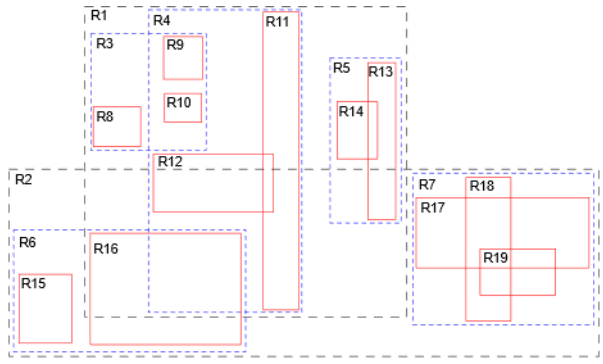




3.3 Árbol-R

Fue propuesta por Antonin Guttman en 1984. El árbol es una estructura de datos en forma de árbol, para construir un árbol R se comienza con un árbol vacío y se van insertando los elementos que queremos que se almacenen.

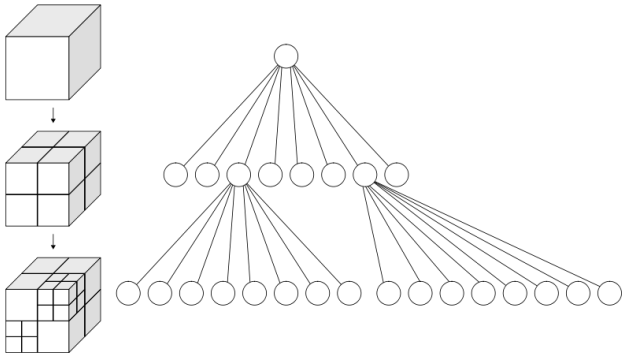
En la estructura del árbol R las hojas son los rectángulos más pequeños que encierran los objetos geométricos que se quieren explorar, la forma de insertar elementos en un árbol R es empezar a recorrer el árbol de la raíz hacia las hojas, en cada paso se identifica de los rectángulos del nivel en cual o en cuales está incluido el rectángulo que voy a insertar, y se recorre hasta llegar a las hojas, el rectángulo se inserta como una hoja en el lugar al que se llegue. Para buscar un rectángulo en el árbol se sigue el mismo procedimiento, pero en lugar de insertar una hoja al final te retorna la hoja que contenga el rectángulo buscado. Este procedimiento se usa para identificar posibles colisiones entre objetos, si luego de la búsqueda no se llega a ninguna hoja es porque el objeto no entra en colisión. [6]



3.4 Octree

Es una estructura de datos jerárquica que descompone el espacio, se usa para objetos que se encuentren en tres

dimensiones. Funciona dividiendo el espacio en ocho cubos iguales, los cuales a su vez son divididos en ocho cubos cada uno. Este proceso de división se continua hasta llegar a la dimensión buscada, si hablamos de colisiones buscaríamos llegar al tamaño del objeto más pequeño o a la distancia que debe existir entre dos objetos para que no colisionen. Se procede a revisar cuales objetos están en el mismo cubo, y estos serían los que tengan riesgo de colisión. [7]



REFERENCIAS

1. Klein, K. Robotic bee could help pollinate crops as real bees decline. *NewScientist*. Recuperado el 26 de Agosto de 2018: https://www.newscientist.com/article/2120832-robotic-bee-could-help-pollinate-crops-as-real-bees-decline/

2. Karumanchi, N. Data Structures and Algorithms Made Easy in JAVA. CareerMonk Publications, Bombay, 2018.

3. MacDonald, T. Spatial Hashing. Gamedev.net. Recuperado el 25 de Agosto de 2018: https://www.gamedev.net/articles/programming/general-and-gameplay-programming/spatial-hashing-r2697/

4. James. Introductory Guide to AABB Tree Collision Detection. AZURE FROM THE TRENCHES. Recuperado el 25 de Agosto de 2018: https://www.azurefromthetrenches.com/introductory-guide-to-aabb-tree-collision-detection/

5. Karumanchi, N. Data Structures and Algorithms Made Easy in JAVA. CareerMonk Publications, Bombay, 2018.

6. Antonin Guttman. 1984. R-trees: a dynamic index structure for spatial searching. In Proceedings of the 1984 ACM SIGMOD international conference on Management of data (SIGMOD '84). ACM, New York, NY, USA, 47-57. DOI=http://ezproxy.eafit.edu.co:2079/10.1145/602259.602266

7. Jane Wilhelms and Allen Van Gelder. 1992. Octrees for faster isosurface generation. ACM Trans. Graph. 11, 3 (July 1992), 201-227. DOI=http://ezproxy.eafit.edu.co:2079/10.1145/130881.130882