**Drones Polinizadores (Como Detectar sus Colisiones)**



Juan Diego Gutierrez Montoya

[jdgutierrm@eafit.edu.co](mailto:jdgutierrm@eafit.edu.co)

Juanita Vanegas Elorza

[jvanegase@eafit.edu.co](mailto:jvanegase@eafit.edu.co)

Mauricio Toro

[mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)

Medellín, Colombia

2018

**Resumen**

El problema consiste en evitar las colisiones entre drones abeja, la solución a este problema es importante ya que las abejas son necesarias para la polinización de muchas especies de cultivos y pueden contribuir a mejorar su rendimiento. Los problemas relacionados son por ejemplo el tiempo de autonomía de los drones o su acceso a todos los tipos de plantas.

**Introducción**

El siguiente documento tratará acerca de la importancia de la polinización con abejas y como ha disminuido en gran medida la población de ellas gracias a la contaminación y destrucción de sus hábitats naturales; al igual de la alternativa en caso de su extinción con el uso de drones abeja que harán sus acciones.

Además, se tratarán los distintos problemas que surgen con esta alternativa como lo podrían ser la autonomía de sus baterías, el acceso a todos los tipos de plantas y el más importante aquí tratado, como evitar las colisiones entre todos estos drones a poca distancia unos de otros.

**Problema**

El problema que se está resolviendo es como evitar las colisiones entre drones abeja, con el fin de no dañar los sistemas o su autonomía. Este problema debe ser resuelto ya que al estar todos los drones tan cerca puede ser un factor importante en la implementación de esta solución para la falta de abejas reales. Se debe tener en cuenta además que el problema debe ser resuelto ya que según estudios un tercio de los alimentos consumidos por los humanos son polinizados por las abejas y la falta de ellas ocasionaría escasez de alimento y quizá la desaparición de muchas especies de cultivos y flores.

**Trabajos Relacionados**

-Árbol dinámico AABB: Como saber cuándo dos cuerpos están colisionando potencialmente. La solución a este problema es usar inserción y balanceo de un árbol binario.

-Hashing espacial: usar una tabla de hash en 3 dimensiones y evitar las colisiones entre los elementos. La solución propuesta es tener presente en todo momento la ubicación espacial de cada uno de los elementos en el espacio.

-Detección de colisiones en el espacio árboles AABB: como detectar colisiones en el espacio de un juego. La solución plateada es el uso de árboles binarios de tipo AABB.

-Quadtree: El problema es que se debe hacer procesamiento de imágenes. La solución implementada es el uso de Quadtrees que son árboles en los cuales cada nodo tiene 4 hijos.

**Palabras clave**

**Estructuras de datos**

02221111111111111

8

3

4

5

6

7

9

n

10

1

2

02221111111111111

3

4

5

6

7

8

9

n

10

1

2221111111111111

X

Y

1625

86

352

1196

736

120

1588

562

247

1253

968

1453

500

340

778

618

727

149

826

890

235

120

489

90

Arreglo simple de n posiciones; cada posición contiene un numero entero que representa la coordenada X o Y de acuerdo al arreglo.

**Operaciones de la estructura de datos**

Arreglo sin ordenar

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3** | **0** | **1** | **8** | **7** | **2** | **5** | **4** | **9** | **6** |

Tomo la posición 0 como pivote y comparo desde la última posición hasta que pivote > comparación.

pivote

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 0 | 1 | 8 | 7 | 2 | 5 | 4 | 9 | 6 |

Intercambio pivote y comparación.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 0 | 1 | 8 | 7 | 3 | 5 | 4 | 9 | 6 |

Cambio el sentido de la comparación desde la primera posición hasta que pivote < comparación.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 0 | 1 | 8 | 7 | 3 | 5 | 4 | 9 | 6 |

Intercambio pivote y comparación.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 0 | 1 | 3 | 7 | 8 | 5 | 4 | 9 | 6 |

Cambio el sentido de la comparación hasta que pivote > comparación. Si ya se comparó con todos y no se cumple la condición pivote ya está en su posición.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 0 | 1 | 3 | 7 | 8 | 5 | 4 | 9 | 6 |

2

1

0

6

9

4

5

8

7

3

El elemento 3 del arreglo ya está ordenado, el arreglo se divide en dos sub arreglos a los cuales se les aplicará el mismo método hasta quedar ordenados todos los elementos.

**Referencias**

<http://www.randygaul.net/2013/08/06/dynamic-aabb-tree/>

<https://www.gamedev.net/articles/programming/general-and-gameplay-programming/spatial-hashing-r2697/>

<https://www.azurefromthetrenches.com/introductory-guide-to-aabb-tree-collision-detection/>

<https://en.wikipedia.org/wiki/Quadtree#Some_common_uses_of_quadtrees>