ALGORITMO PARA PREVENIR COLOCIONES ENTRE ABEJAS ROBOTICAS

Juan Camilo Guerrero Alarcón
Universidad Eafit
Colombia
jcguerrera@eafit.edu.co

Santiago Pulgarín Vásquez
Universidad Eafit
Colombia
spulgarinv@eafit.edu.co

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

RESUMEN

El principal problema radica en que no se tiene un algoritmo implementado para detectar las abejas robóticas y que estas no se colisionen, es demasiado importante para que no disminuya la cantidad de abejas, saber dónde eran y a que objetivo tienen que llegar sin ningún problema, por el momento existen problemas similares como lo son Quadtree, AABB Tree entre otros.

1. INTRODUCCIÓN

Gran parte de cultivos en todo el mundo necesitan de un proceso llamado polinización el cual como es bien sabido es realizado por abejas y otros insectos. Por algunos fenómenos climáticos las abejas han reducido en gran cantidad lo cual está siendo un serio problema para los agricultores, lo que nos lleva a pensar si se podrían implementar una especie de RoboBees los cuales podrían ayudar con el problema e implementarlos en un futuro no muy lejano y a su vez mejorar la calidad de los cultivos

2. PROBLEMA

La implementación de estas abejas resulta ser una buena manera para este caso, pero con dicha implementación se necesitará una forma y algoritmo eficiente para poder prevenir choques o colisiones entre ellas para que se desarrolle el proceso en dichos cultivos con eficiencia.

3. TRABAJOS RELACIONADOS

3.1 Spatial Hashing

Consiste en la falta de forma simple de reducir la cantidad de objetos probados durante la detección de colisiones, ya que los métodos tradicionales que se venían implementando no cumplen lo requerido y requieren demasiado tiempo.

Una posible solución es pensar en un hash espacial es una extensión de 2 o 3 dimensiones de la tabla hash, La idea básica de una tabla hash es que tomes un dato (la 'clave'), lo ejecutes a través de alguna función (la 'función hash') para producir un nuevo valor (el 'hash'), y luego usar el hash como un índice en un conjunto de ranuras ('cubos').

Para almacenar un objeto en una tabla hash, ejecuta la clave a través de la función hash y almacena el objeto en el depósito al que hace referencia el hash. Para encontrar un objeto, ejecuta la tecla a través de la función hash y busca en el depósito al que hace referencia el hash. [1]

3.2 AABB Tree Collision Detection

Radica básicamente en que no se tiene la maneja más optima de encontrar o detectar colisiones en el mundo a todo lo que se mueva y los métodos actuales no han dado los mejores resultados para que esto empiece a tomar cartas en el asunto, y en este problema se considera como un algoritmo implementado en un videojuego.

La solución que se ha revisado trata de Los AABB, son más simples de lo que parecen: son esencialmente cajas cuyos ejes (así x, y para 2d y x, y, z para 3d) se alinean / corren en la misma dirección. La parte delimitadora del nombre se debe a que, cuando se usan para la detección de colisiones o como parte de un árbol, comúnmente contienen o unen otras casillas. [2]

3.3 Quadtree

Un quadtree es una estructura de datos de árbol en la que cada nodo interno tiene exactamente cuatro hijos. Los Quadtrees son el análogo bidimensional de octrees y se usan con mayor frecuencia para dividir un espacio bidimensional subdividiéndolo recursivamente en cuatro cuadrantes o regiones. Los datos asociados con una célula de la hoja varían según la aplicación, pero la célula de la hoja representa una "unidad de información espacial interesante".

Las regiones subdivididas pueden ser cuadradas o rectangulares, o pueden tener formas arbitrarias. Esta estructura de datos fue nombrada quadtree por Raphael Finkel y JL Bentley en 1974. Una partición similar también se conoce como Q-tree. [3]

3.4 Dynamic AABB Tree

Un árbol dinámico de AABB es un árbol de búsqueda binario para la partición espacial. Este árbol cuenta con unas propiedades para implementar y que lo hace más optimo:

- Insertar
- Retirar

- Actualizar

El nodo del árbol AABB puede construirse cuidadosamente para ocupar un espacio mínimo, ya que los nodos están siempre en uno de dos estados: ramas y hojas. Como los nodos se almacenan en una matriz, esto permite que los nodos sean referenciados por índice integral en lugar de puntero. Esto permite que la matriz interna crezca o se reduzca según sea necesario sin temor a dejar punteros colgantes en cualquier lugar.

La idea del árbol es permitir que los datos del usuario se almacenen solo dentro de los nodos hoja. Todos los nodos de sucursales contienen solo una AABB que envuelve a ambos de sus hijos. [4]

REFERENCIAS.

[1] Macdonald, T. Spatial Hashing. GameDev.net, 2018. https://www.gamedev.net/articles/programming/general- and-gameplay-programming/spatial-hashing-r2697/.

[2] Introductory Guide to AABB Tree Collision Detection – Azure From The Trenches. Azurefromthetrenches.com, 2018. https://www.azurefromthetrenches.com/introductory- guide-to-aabb-tree-collision-detection

[3]Quadtree.En.wikipedia.org,2018.https://en. wikipedia.org/wiki/Quadtree.

[4] GAUL, R. Dynamic AABB Tree Gaul, R. Dynamic AABB Tree. Randy Gaul's Game Programming Blog, 2018.