PONTIFICIA UNIVERSIDAD JAVERIANA

Documentación: Proyecto Curiosity# 1

SISTEMAS OPERATIVOS

INTEGRANTES:

Santiago Vera
Nicolas Arciniegas
Daniela Torres

GRUPO # 8

PROFESOR:
John Corredor Franco

1 Especificaciones del proyecto

1.1 Introducción

Para la representación en el plano espacial del movimiento de un robot se puede utilizar distintas estructuras de datos y algoritmos útiles que permite entender la ubicación y movimiento de un cuerpo, en este caso, que se mueve en dos dimensiones, además de que encontramos uno de los más famosos problemas dentro de la computación, el problema del camino más corto. En este trabajo veremos como los grafos nos permiten entender el plano de coordenadas ordenado.

1.2 Objetivos del proyecto

Entender de qué manera se ubican objetos dentro de un plano en dos dimensiones con diferentes estructuras de datos para últimamente ser capaz de encontrar la ruta de menos esfuerzo dado la ubicación de los puntos, así como, ser capaz de ordenar jerárquicamente diferentes puntos en el plano ordenado. Esto con el propósito de entender diferentes algoritmos como el backtracking y el deep first search, así como, el concepto de recursividad con un ejemplo práctico.

1.3 Propósito del sistema

Permitir al usuario efectuar movimientos y agregar elementos para similar el escenario de un robot moviéndose que se encuentra con posibles colisiones por los objetos, asi como, ser capaz de organizar estos objetos jerárquicamente por medio de un Kd-Tree para ser capaz de encontrar una manera de pasar por los objetos puestos por el usuario

2 Diseño del proyecto

2.1 Tokenizacion

Se ha hecho una terminal interactiva para el usuario en la que él podrá escribir diferentes comandos que son fáciles de entender y que describen la acción por sí sola

```
agregar_movimiento Avanzar 12 cm
agregar_movimiento Girar 25 grados
```

para agregar un movimiento

```
agregar_elementos Roca 12 cm 34 78
```

y para agregar un análisis

```
agregar_analisis Perforar Roca 12 cm el comentario del usuario
```

esto permitirá al sistema agregar comandos y movimientos al sistema
asi mismo puede cargar comandos y elementos de la siguiente manera

```
cargar_elementos ArchivoDeTexto.txt
cargar_comando ArchivoDeTexto.txt
```

2.2 Conversión de unidades

Para utilizar el Sistema Internacional de Unidades, un prompt válido para las unidades del curiosity es mm, cm y km, se ha creado en el archivo conversionUnidades.cpp el manejo de estas unidades de medida ,así como, una validación para no aceptar valores puesto por el usuario que no son números o números negativos.

2.3 simular comando

El programa debe ser capaz de simular como sería el movimiento del curiosity dado una orden de movimientos ordenados para el movimiento paso por paso, así como detectar colisiones con los elementos puestos por el usuario.

El programa pasará por cada movimiento y ejecutará una función que puede encontrar los componentes en ejes x y en eje y del curiosity, tomando la función seno y la función coseno de los grados según corresponda y tomando cuanto desea avanzar la persona para simular el movimiento en el plano cartesiano, por cada movimiento, vera si una línea, que va del punto al que llegó al centro de cada elemento es más pequeño que la mitad del lado de un elemento que se representa como un cuadrado, si la línea del punto en donde esta el curiosity al centro del elemento es más pequeña que la mitad del lado del elemento habrá una colisión.

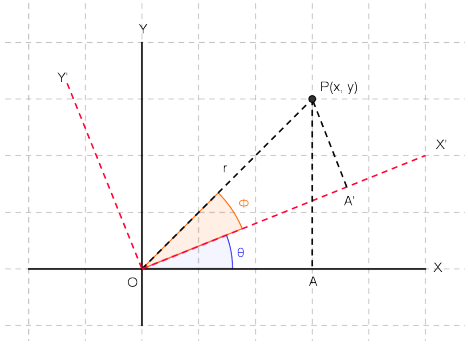

Figure 1: entendemos movimiento como una suma vectorial

2.4 Ubicar Elementos

Para ser capaz de ordenar jerárquicamente los elementos puestos por el usuario, hemos diseñado un Kd-Tree, un Kd-Tree permite separar desde un elemento cualquiera, para ello se rquiere subdividir el espacio en dos partes, encuentra otro punto y vuelve a subdividir dos espacios el Kd-Tree en dos dimensiones funciona de la siguiente manera:

Ubicamos los elementos en un espacio y creamos un corte Pasando por los

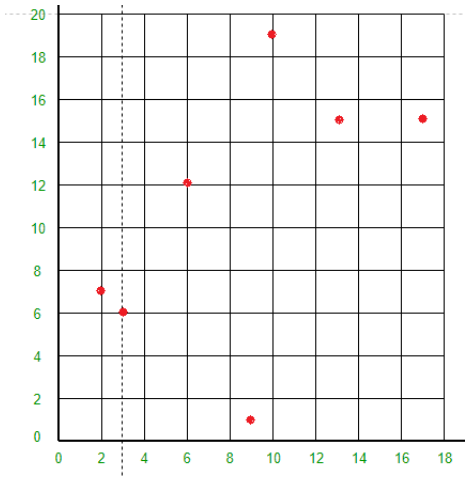

Figure 2: El programa encontrará un elemento y separa el espacio en el eje x.

puntos en las coordenadas x y y, seleccionara según el menor, y el mayor así como, si es en coordenada x y y.

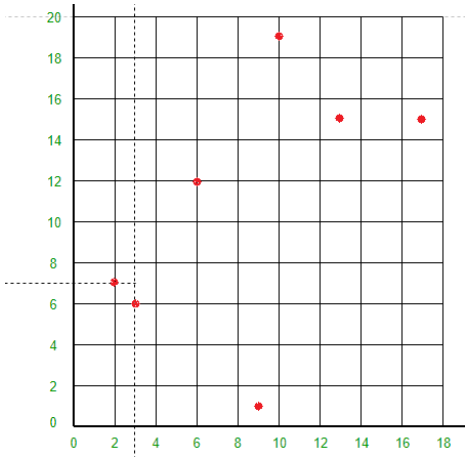

Figure 3: subdivisión en x

El programa encuentra un elemento en el subespacio a la derecha y a la izquierda en el eje x del elemento y hace cuadrantes en el eje y

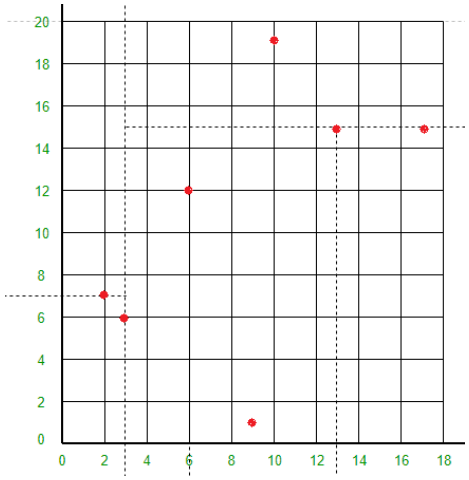

Figure 4: Subdivisión pasando al eje y

El programa volverá a tomar valores en el eje x, y creará más subespacios

De esta manera se irá organizando jerárquicamente los elementos, cada uno con coordenada x y y

2.5 Determinar un cuadrante

Dentro de la implementación del kd-tree, se agregara una función en cuadrante que atravesará el kd-tree buscando solo los puntos que se encuentren en un cuadrante con una longitud proporcionada por el usuario y verificará por cada elemento con coordenadas x y y. si este se encuentra dentro de los límites propuestos por el usuario atravesara todos los puntos

3 El problema de la ruta mas corta

Para encontrar la solución al problema de la ruta mas corta hemos encontrado los siguientes pasos

3.1 Ordenar los elementos

Para ello hemos diseñado un árbol balanceado que toma el valor medio, va creando ramas tal que desde este valor medio podamos llegar a todos los vecinos y seleccionamos la raiz como el valor medio indicado para empezar a hacer el algoritmo

Figure 5: Ordenamiento en el espacio bidimensional

3.2 Estimar las distancias de un punto

Debemos, tomando el punto que seleccionamos estimar las distancias de todos los puntos, encontramos la menor distancia y creamos un camino entre el nodo inicial y el nodo que encontramos

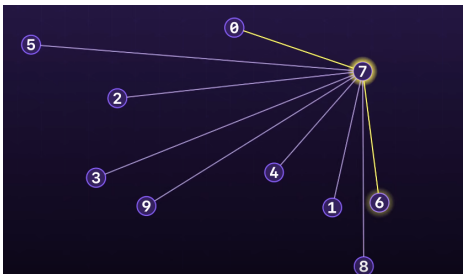

Figure 6: Estimación de distancias desde un punto

3.3 Generar un camino

Recursivamente generaremos un camino, pasando por el nodo que encontrando, estimando las distancias y tomando la menor distancia hasta volver al nodo que seleccionamos como nodo inicial

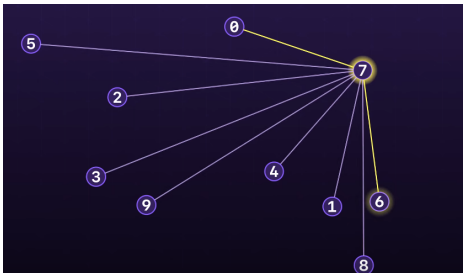

Figure 7: Estimación de distancias desde un punto

3.4 Prevención de ciclos

El sistema mantiene la información de los nodos que ya hay visitado para no seleccionarlos como posibles distancias a las que ir

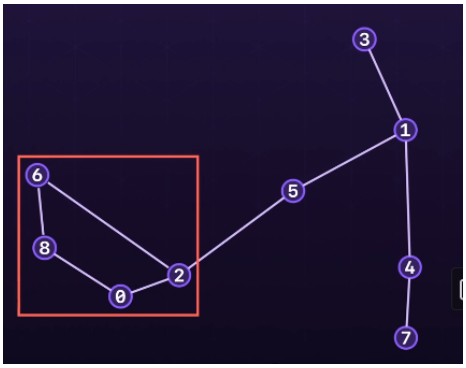

Figure 8: Ejemplo de una ruta que posee un ciclo