

José Alfredo Quevedo Suarez, Santiago Herrera Parra, Antonio Garay Pinzón

Guía Técnica

Este juego implementa un **RTS (Real-Time Strategy)** con hormigas.

Está sustentado por una serie de **estructuras de datos tipo grafo**, acompañadas de:

- Arrays dinámicos,
- Objetos literales de configuración,
- Clases modelo (Node, Ant),
- Objetos globales que representan el **estado del juego**.

Toda la lógica se orquesta desde el objeto global game.

Estructuras de Datos Principales

SPRITES (Pixel Art)

```
const SPRITES = { worker: [ ... ], soldier:[ ... ], ... }
```

Tipo:

- **Objeto literal** que contiene **matrices bidimensionales (7×7 o 3×3)**.
- Cada sprite es un Array<Array<number>>.

Rol:

- Representa visualmente unidades y edificios.
- Cada número indica un color (1=color del equipo, 2=negro, 3=blanco, 4=gris).

Importancia estructural:

- Abstracción gráfica: el sistema de dibujo **solo necesita una clave ('worker', 'tank', 'farm')**, no las formas específicas.

TEAMS

```
const TEAMS = { fire:{c:'#e74c3c'}, earth:{c:'#2ecc71'}, ... }
```

Tipo:

- **Objeto literal con sub-objetos**.

Rol:

- Guarda los colores oficiales de cada civilización.

COSTS, STATS

```
const COSTS = { worker:{l:3,w:3,f:0}, farm:{l:0,w:10,f:0}, ... }
```

```
const STATS = { worker:{hp:40,dmg:2,spd:1.5}, ... }
```

Tipo:

- **Objetos literales** con sub-objetos que definen:
 - Costos de construcción,
 - Estadísticas de combate de unidades.
-

Estructura del Estado Global (game)

```
const game = {  
  nodes: [], ants: [], fx: [],  
  res:{ player:{l:50,w:50,f:20}, cpu1:..., ... },  
  cam:{ x:0,y:0,zoom:1,drag:false,lx:0,ly:0 },  
  ...  
}
```

Tipo:

- **Objeto gigante de estado global** que contiene varias estructuras internas:

Contenidos clave:

Propiedad	Tipo	Función
nodes	Array de Node	Representa habitaciones — nodos del grafo
ants	Array de Ant	Todas las hormigas en el mapa
fx	Array de efectos visuales	Animaciones temporales
res	Objeto → objetos	Recursos por facción
cam	Objeto	Estado de la cámara
tool, attackMode, attackTarget	Variables de control	Estado del modo de interacción

Clases y Estructuras Asociadas

Clase Node → Nodos del Grafo

```
class Node {  
    constructor(x,y,type,team,col) {  
        this.x; this.y;  
        this.type; this.team; this.col;  
        this.conns = [];  
        this.res = {l:0,w:0,f:0};  
        this.queue = [];  
    }  
}
```

Tipo:

- **Instancias de clase** almacenadas en game.nodes.

Campos importantes:

- conns: **Array de referencias directas** → estructura de grafo no dirigido.
- res: Objeto con recursos que produce el nodo.
- queue: **Array de producción** de hormigas:
- [{ type:'soldier', ready: 174958284 }]

Esto acaba en un grafo dinámico, donde los nodos son habitaciones y las aristas son túneles.

Importante:

Cuando se borra un nodo se actualizan todas las otras listas conns, manteniendo la integridad del grafo.

Clase Ant → Unidades Móviles

```
class Ant {  
    constructor(type,node,team) {  
        this.type;  
        this.node; // Referencia a Node  
        this.target; // Node destino
```

```
    this.team;  
  
    this.stats...  
  
}  
  
}
```

Tipo:

- Objetos dinámicos agregados a game.ants.

Relación con nodos:

node y target son **referencias a instancias específicas** de Node.
Esto permite que:

- Las hormigas detecten enemigos en el mapa,
- Se desplacen por nodos conectados,
- Busquen rutas (simple pathfinding local).

Efectos visuales (FX)

```
this.fx.push({x:a.x, y:a.y, t:15, type:'dead'});
```

Tipo:

- **Array de objetos literales:** {x, y, t, type}.

Uso:

- Se reduce t cada frame hasta desaparecer.

Graficado con Canvas

```
ctx.translate(...)
```

```
ctx.scale(...)
```

```
Node.draw();
```

```
Ant.draw();
```

Los nodos y hormigas se basan exclusivamente en:

- Matrices de sprites,
- Coordenadas absolutas,
- Transformaciones del canvas.

Lógica de Conexiones (Grafo del Hormiguero)

Los nodos están enlazados así:

```
node.conns.push(otherNode);
```

```
otherNode.conns.push(node);
```

Tipo:

- **Estructura de grafo no dirigido con lista de adyacencia.**

Funciones importantes:

- connect(n)
- disconnect()
- IA y hormigas consultan conexiones para moverse:
- `this.node.conns[Math.floor(Math.random()*this.node.conns.length)]`

IA y Estructuras de Datos

El sistema de IA usa:

Recursos:

```
this.res[team] = {l,w,f}
```

Selección de nodos:

```
let nodes = this.nodes.filter(n => n.team===team);
```

Expansión de base:

- Busca nodos cercanos,
- Crea nodos nuevos,
- Conecta a nodos existentes.

Entrenamiento:

- Agrega objetos al array queue.

Sistema de Recursos

Cada facción tiene:

```
res:{ player:{l,w,f}, cpu1:{}, cpu2:{}, cpu3:{} }
```

Cada nodo tiene recursos internos:

```
node.res = {l,w,f}
```

Los trabajadores:

- Extraen de node.res,
- Lo llevan a game.res[team].

Minimapa

```
m.fillRect((node.x+off)*s, (node.y+off)*s, 3, 3);
```

Tipo:

- Render en un canvas separado representando nodos como píxeles.

Síntesis de Todas las Estructuras de Datos

Tipo

Explicación

Grafo (lista de adyacencia) Nodos con conexiones recíprocas

Array dinámico de objetos Cada hormiga enlazada a un Node

Array (cola) por nodo Construcción de hormigas

Array de objetos Animaciones temporales

Objetos → Matrices Pixel art de entidades

Objetos literales Colores y facciones

Objeto → Sub-objetos Hojas, agua, comida por facción

Objeto Recursos internos del edificio

Arrays de nodos filtrados Toma decisiones por estructura del grafo