Cyberpunk 2077 Skill Tree Planner

Documentación del Código y Algoritmos

Geor Sebastián Gómez Correa
Tomás Camilo García López
Eduardo Castellanos Márquez

UNIVERSIDAD NACIONAL DE COLOMBIA

Manual Técnico del Desarrollador

1. Arquitectura del Sistema

El proyecto sigue una arquitectura **JAMstack (JavaScript, APIs, Markup)** simplificada, sin base de datos en servidor. Todo el procesamiento de datos pesado se realiza "Offline" (Python) y la lógica de negocio corre en el cliente (Browser).

Estructura de Archivos

- **Backend (Pre-procesamiento):**
  - `process_data.py`: Script ETL (Extract, Transform, Load). Convierte hojas de cálculo (`.xlsx`) en un grafo JSON consumible.
- **Frontend (Cliente):**
  - `index.html`: Estructura semántica y contenedores del DOM.
  - `style.css`: Motor de renderizado visual (Grid, Flexbox, Posicionamiento Absoluto, SVG Styling).
  - `app.js`: Controlador principal. Maneja eventos, estado y renderizado.
- **Capas de Datos (Persistencia):**
  - `skills_data.js`: Base de datos estática (Solo lectura). Define el grafo.
  - `layout_config.js`: Coordenadas X/Y de cada nodo (UI).
  - `presets_data.js`: Diccionario de configuraciones guardadas (Builds).

2. Backend: Procesamiento de Datos (`process_data.py`)

Este script resuelve el problema de la **integridad referencial** del grafo antes de que llegue al navegador.

Funciones Clave

1. `clean_split(value)`:
    - **Problema:** El Excel contiene datos sucios (`NaN`, `"0"`, `"22.0"`).
    - **Solución:** Sanitiza los strings, elimina decimales flotantes y convierte listas separadas por comas en Arrays de Python.
2. `find_icon_path(tree, title)`:
    - **Lógica:** Normalización de strings. Convierte *"Run 'N' Gun"* a `runngun` y busca coincidencias en la carpeta de assets para asignar la ruta de la imagen automáticamente.
3. **Inferencia de Padres (Reverse Engineering):**
    - **Algoritmo:** El Excel original es una *Lista de Adyacencia Unidireccional* (Solo conoce hijos: `next perk ID`).
    - **Proceso:** El script itera sobre todos los nodos. Si el Nodo A dice que su hijo es B, el script busca al Nodo B y le inyecta "A" en su lista de `parents`.
    - **Resultado:** Convierte el árbol en un grafo bidireccional navegable.

---

3. Frontend: Controlador Principal (`app.js`)

Es el cerebro de la aplicación. Gestiona el ciclo de vida de la aplicación.

A. Gestión de Estado (`GLOBAL_STATE`)

A diferencia de aplicaciones simples que leen el DOM, aquí mantenemos una **"Single Source of Truth"** en memoria.

- Al iniciar, se clona profundamente (`JSON.parse(JSON.stringify)`) el objeto `CYBERPUNK_DATA` hacia `GLOBAL_STATE`.
- Cualquier cambio (comprar/vender) modifica `GLOBAL_STATE`.
- El DOM se borra y se repinta basado en `GLOBAL_STATE` (Reactividad manual).

B. Ciclo de Renderizado (`renderTree`)

1. **Limpieza:** `innerHTML = ''` del contenedor.
2. **Generación de Nodos:** Itera sobre los datos del árbol activo.
    - Crea el `<div>` con la clase según su tipo (`major`, `minor`, `ultimate`).

    - Asigna la imagen de fondo (`getTileImage`) y el icono.
    - Aplica coordenadas CSS `top/left` desde `layout_config.js`.
3. **Dibujado de Conexiones:** Se llama a `drawConnections()` al final, con un `setTimeout(0)` para asegurar que el navegador ha calculado las posiciones geométricas de los nuevos `divs`.

C. Sistema de Eventos (Interacción)

`handleLeftClick(node)` (Comprar)

Implementa una máquina de estados para la transacción:

1. **Validación:**
    - ¿`availablePoints > 0`?
    - ¿El nodo es `available` O (`selected` Y `current < max`)?
2. **Mutación:** Aumenta `currentLevel`.
3. **Trigger de Desbloqueo:** Si `currentLevel == maxLevel`, dispara `checkUnlock()` sobre sus hijos.

`handleRightClick(node)` (Vender)

Implementa la lógica de protección topológica:

1. **Validación de Integridad:**
    - Si el nodo está al máximo (`maxLevel`), escanea sus hijos (`children`).
    - Si algún hijo tiene `currentLevel > 0`, **abortar transacción**. Esto evita "islas flotantes" de habilidades activas sin conexión a la raíz.
2. **Mutación:** Disminuye `currentLevel`.
3. **Trigger de Bloqueo:** Si el nodo baja del máximo, dispara `checkLock()` sobre sus hijos para volverlos rojos (`blocked`).

---

4. Algoritmos Gráficos (SVG)

La función `getCircuitPath(x1, y1, x2, y2)` es responsable de la estética "Tech".

Algoritmo "Manhattan Chamfer"

El objetivo es dibujar una línea entre dos puntos evitando diagonales directas que crucen otros elementos.

Entradas: Coordenadas de borde (Bottom del Padre, Top del Hijo). **Lógica:**

1. Calcular `midY` (Punto medio vertical).
2. Dibujar línea vertical desde el Padre hasta `midY - offset`.

3. Dibujar línea diagonal (45°) de longitud `offset`.
4. Dibujar línea horizontal hasta la coordenada X del Hijo (`x2 - offset`).
5. Dibujar línea diagonal inversa (45°).
6. Dibujar línea vertical final hacia el Hijo.

Resultado: Una forma de "escalera" o "S" rígida, típica de diagramas de circuitos electrónicos.

---

5. Sistema de Presets (Importar/Exportar)

Serialización Diferencial

Para no crear archivos de guardado gigantescos, no guardamos todo el objeto del nodo.

- **Exportar:** Solo se guardan los pares `{ ID: Nivel }` de los nodos que han sido modificados (`currentLevel > 0`).
- **Importar:**
    1. `resetInternalState()`: Pone todo a cero.
    2. Itera el JSON del preset y aplica los niveles.
    3. **Recálculo de Estados (`recalculateAllStates`):** Como inyectamos datos "crudos", el grafo no sabe qué nodos deben estar disponibles (azules). Esta función recorre el grafo **3 veces** (para propagar dependencias profundas) verificando:
        - `SI (Padres están Max)` -> `Estado = Available`.
        - `SI (Tiene Puntos)` -> `Estado = Selected`.

---

6. CSS Avanzado (Estilos Dinámicos)

- **Capas SVG:** Los cables se dibujan usando 3 rutas SVG superpuestas (`.cable-bg`, `.cable-separator`, `.cable-core`) para simular un cable plano físico con un núcleo de datos brillante.
- **Backdrop Filter:** Uso de `backdrop-filter: blur(10px)` en el contenedor principal para crear el efecto de "cristal esmerilado" sobre la imagen de fondo, mejorando la legibilidad sin ocultar el arte.
- **Clip-Path:** Uso de polígonos CSS en los botones (`polygon(0 0, 100% 0, 100% 70%...)`) para crear formas irregulares futuristas que no son posibles con `border-radius`.
