

# Guía: Generación Procedural del Mapa (Sin archivo .txt)

Esta guía describe cómo implementar el **Paso 1** de la migración hacia Arena Survival: reemplazar el mapa .txt por generación procedural en memoria y aumentar el tamaño del mundo.

---

## Objetivo

- Eliminar la dependencia del archivo `world01_1.txt` para la carga del mapa.
  - Generar el mapa directamente en memoria con un patrón aleatorio simple.
  - Aumentar el tamaño del mundo de **50×50** a **100×100** tiles para dar sensación de inmensidad.
- 

## Archivos a modificar

Archivo	Cambio
<code>src/main/PanelJuego.java</code>	Aumentar <code>maxWorldcol</code> y <code>maxWorldfilas</code> a 100
<code>src/tiles/TileManager.java</code>	Eliminar <code>cargarMapa()</code> , agregar <code>generarMapaProcedural()</code>
<code>src/entidad/Jugador.java</code>	Ajustar posición inicial de spawn al centro del nuevo mundo

---

## Referencia rápida de tiles

Según `res/tiles/rutaTiles.txt`, estos son los tiles disponibles:

Índice	Colisión	Uso sugerido
0-16	1 (sólido)	Obstáculos (muros, rocas, árboles)
17	0 (libre)	Pasto base principal
18-27	0 (libre)	Pasto alternativo / decoración
28-29	1 (sólido)	Obstáculos adicionales
30-39	0 (libre)	Suelo alternativo
40-42	1 (sólido)	Obstáculos grandes

**Para la generación procedural usaremos:** - 17 como tile base (pasto, sin colisión) - 0 como tile obstáculo (con colisión) - 30 como pasto alternativo (sin colisión, para variedad visual)

## Paso 1: Aumentar el tamaño del mundo

### Archivo: `src/main/PanelJuego.java`

Busca las líneas actuales (líneas 37-38):

```
// ajuste del mundo
public final int maxWorldcol = 50;
public final int maxWorldfilas = 50;
```

Cámbialas a:

```
// ajuste del mundo (arena grande)
public final int maxWorldcol = 100;
public final int maxWorldfilas = 100;
```

[!NOTE] Las variables `maxWorldAncho` y `maxWorldAlto` (líneas 39-40) se recalculan automáticamente porque dependen de `maxWorldcol` y `maxWorldfilas`:

```
public final int maxWorldAncho = maxWorldcol *
    tamañoTile; // 100 * 64 = 6400
public final int maxWorldAlto = maxWorldfilas *
    tamañoTile; // 100 * 64 = 6400
```

No necesitas tocar esas líneas.

## Paso 2: Ajustar posición de spawn del jugador

**Archivo: src/entidad/Jugador.java**

El jugador actualmente aparece en la posición (24, 20) del mundo de 50×50. Con un mundo de 100×100, debemos moverlo al **centro** para que tenga espacio en todas las direcciones.

Busca en el método `setValorePorDefecto()` (líneas 61-67):

```
public void setValorePorDefecto() {  
    worldx = pj.tamanoTile * 24;  
    worldy = pj.tamanoTile * 20;
```

Cámbialo a:

```
public void setValorePorDefecto() {  
    worldx = pj.tamanoTile * (pj.maxWorldcol / 2); // centro  
           horizontal  
    worldy = pj.tamanoTile * (pj.maxWorldfilas / 2); // centro  
           vertical
```

[!IMPORTANT] Esto asegura que el spawn siempre sea el centro del mundo, sin importar el tamaño futuro.

---

## Paso 3: Generación procedural en TileManager

**Archivo: src/tiles/TileManager.java**

### 3.1 Eliminar el método cargarMapa

Elimina completamente el método `cargarMapa(String rutaFile)` (líneas 90-122). Este método ya no será necesario.

También puedes eliminar estos imports que ya no se usarán (líneas 6-8):

```
import java.io.BufferedReader;    // ya no se usa (cargarMapa
    eliminado)
import java.io.InputStream;      // ya no se usa
import java.io.InputStreamReader; // ya no se usa
```

[!NOTE] El método `getImagenTile()` sí usa `InputStream` y `BufferedReader`, así que **NO los elimines** si no estás seguro. Revisa que `getImagenTile` aún los necesite. En este proyecto **sí los necesita**, así que **déjalos**.

## 3.2 Modificar el constructor

Reemplaza la llamada a `cargarMapa(...)` por `generarMapaProcedural()`:

**Antes** (líneas 35-40):

```
public TileManager(PanelJuego pj) {
    this.pj = pj;
    mapaPorNumeroTile = new int[pj.maxWorldcol][pj.maxWorldfilas];
    getImagenTile("/tiles/rutaTiles.txt");
    cargarMapa("/mapas/world01_1.txt");
}
```

**Después:**

```
public TileManager(PanelJuego pj) {
    this.pj = pj;
    mapaPorNumeroTile = new int[pj.maxWorldcol][pj.maxWorldfilas];
    getImagenTile("/tiles/rutaTiles.txt");
    generarMapaProcedural(); // <-- reemplaza cargarMapa
}
```

## 3.3 Implementar `generarMapaProcedural()`

Agrega este método nuevo en `TileManager`, después del constructor:

```
/**
 * Genera un mapa procedural en memoria con pasto base y
 * obstáculos aleatorios.
 * Reemplaza la carga desde archivo .txt.
 */
private void generarMapaProcedural() {
    int baseTile = 17;           // pasto base (colisión = 0)
    int baseTileAlt = 30;        // pasto alternativo (colisión = 0)
    int obstacleTile = 0;        // obstáculo (colisión = 1)
```

```

// Centro del mundo: zona segura de spawn
int centroCol = pj.maxWorldcol / 2;
int centroFila = pj.maxWorldfilas / 2;
int radioSeguro = 5; // radio de tiles libres alrededor del
    spawn

for (int col = 0; col < pj.maxWorldcol; col++) {
    for (int fila = 0; fila < pj.maxWorldfilas; fila++) {

        // Zona segura alrededor del spawn (sin obstáculos)
        int distCol = Math.abs(col - centroCol);
        int distFila = Math.abs(fila - centroFila);
        if (distCol <= radioSeguro && distFila <=
radioSeguro) {
            mapaPorNumeroTile[col][fila] = baseTile;
            continue;
        }

        // Bordes del mundo = obstáculos sólidos (muralla
natural)
        if (col == 0 || col == pj.maxWorldcol - 1
|| fila == 0 || fila == pj.maxWorldfilas - 1) {
            mapaPorNumeroTile[col][fila] = obstacleTile;
            continue;
        }

        // Interior: distribución aleatoria
        double random = Math.random();
        if (random < 0.05) {
            // 5% obstáculos
            mapaPorNumeroTile[col][fila] = obstacleTile;
        } else if (random < 0.20) {
            // 15% pasto alternativo (variedad visual)
            mapaPorNumeroTile[col][fila] = baseTileAlt;
        } else {
            // 80% pasto base
            mapaPorNumeroTile[col][fila] = baseTile;
        }
    }
}
}

```

[!TIP] **Zona segura:** El `radioSeguro = 5` crea una zona de 11×11 tiles sin obstáculos alrededor del spawn, evitando que el jugador quede atrapado al iniciar.

[!TIP] **Bordes del mundo:** Los bordes son obstáculos sólidos para que el jugador no salga del mapa y evitar `ArrayIndexOutOfBoundsException` en `detectorColisiones`.

---

## Paso 4: Eliminar import innecesario (opcional)

En `TileManager.java`, la línea 9 tiene un import sin usar:

```
import java.nio.Buffer; // <-- eliminar, no se usa en ningún lado
```

---

## Resumen de cambios

`src/main/PanelJuego.java`

- └ `maxWorldcol: 50 → 100`
- └ `maxWorldfilas: 50 → 100`

`src/entidad/Jugador.java`

- └ `spawn: posición fija (24,20) → centro dinámico del mundo`

`src/tiles/TileManager.java`

- └ Eliminar método `cargarMapa()`
  - └ Constructor: `cargarMapa("...") → generarMapaProcedural()`
  - └ Nuevo método: `generarMapaProcedural()`
  - └ (Opcional) Eliminar `import java.nio.Buffer`
- 

## Paso 5: Verificación

1. **Compilar y ejecutar** el juego.
2. **Verificar que el mundo sea grande:** camina en cualquier dirección y notarás mucho más espacio que antes.
3. **Verificar colisiones:** camina hacia un tile de obstáculo (tile 0) y confirma que el jugador **no lo atraviesa**.
4. **Verificar spawn seguro:** al iniciar, el jugador debe estar rodeado de pasto libre sin obstáculos inmediatos.
5. **Verificar bordes:** camina hacia los bordes del mapa y confirma que hay un muro de obstáculos que impide salir.

## Posibles errores y solución

Error	Causa	Solución
ArrayIndexOutOfBoundsException en detectorColisiones	Jugador se mueve fuera de los límites del array	Los bordes sólidos evitan esto. Si persiste, agregar chequeo de límites en <code>chektile()</code>
Jugador no puede moverse	El <code>baseTile</code> (17) tiene colisión 1	Verifica en <code>rutaTiles.txt</code> que la línea 18 diga <code>/tiles/tile_17.png;0</code>
FPS bajos	Mundo demasiado grande	Reducir a 100×100 (ya está). El sistema de cámara en <code>draw()</code> solo renderiza tiles visibles

## Siguiente paso

Una vez verificado que todo funciona, el siguiente avance será: - Introducir **ruido de Perlin** para generar biomas (zonas de bosque, desierto, piedra). - Implementar **spawners de enemigos** en los bordes del área visible. - Agregar **variación procedural por regiones** para hacer el mundo menos repetitivo.