Generación Procedural de Mapas 2D usando BSP en C++

Matías Aitel Peters Valdivia

29 de junio de 2025

Resumen

En este informe busco presentar una implementación en C++ de la técnica *Binary Space Partitioning* (BSP) para la generación de mapas 2D, inspirada en la estructura de niveles del videojuego *Hades*. Se contextualiza la generación procedural de contenido (PCG), se estudian otras técnicas relacionadas, y se describe en detalle la metodología de implementación y experimentación, así como los resultados obtenidos.

Índice

1. Introducción

La Generación Procedural de Contenido (PCG) permite crear contenido de forma automática que beneficia la rejugabilidad, personalización y eficiencia en el desarrollo de videojuegos. El juego *Hades*, del estudio Supergiant Games, se caracteriza por su estructura *rogue-like*, donde cada partida genera una secuencia única de *cámaras* de combate y *pasillos* de transición.

Para replicar esta estructura utilizé el algoritmo BSP, una técnica de división recursiva del espacio utilizada en renderizado y generación de niveles. El BSP permite subdividir un mapa 2D en regiones rectangulares que serian como "hojas", estas pueden albergar habitaciones. Posteriormente, estas se conectan con pasillos para formar la estructura completa del nivel.

Me convencí de utilizar BSP por su simplicidad, adaptabilidad a mapas en consola y su capacidad para generar niveles variados con estructuras no lineales, tal como ocurre en Hades, aunque dada la dimension del juego como tal, obviamente se simplicicó el concepto para que calce con lo pedido.

2. Estado del Arte

Diversas técnicas de PCG han sido exploradas en videojuegos:

- **Drunkard's Walk**: simulación de caminantes aleatorios para explorar el mapa (usado en algunos roguelikes clásicos).
- **Wave Function Collapse (WFC)**: algoritmo de restricción basado en patrones, popular por generar estructuras coherentes y detalladas.
- **Técnicas de grafos**: para conectar nodos representando habitaciones.

Videojuegos como *Dead Cells* y *The Binding of Isaac* emplean distintas formas de PCG para asegurar variedad y rejugabilidad. Hades destaca por ofrecer mapas estructurados como secuencias de cámaras con decisiones ramificadas.

BSP se ha usado tradicionalmente en motores como Quake para optimización de renderizado, pero también es eficaz para dividir espacios lógicos con el diseño de niveles

3. Metodología

Se implementó BSP en C++ utilizando estructuras `Room` y `BSPNode`, las cuales representan habitaciones y divisiones del espacio. El mapa se genera como una matriz 2D de caracteres.

3.1. Parámetros

- `MAP_WIDTH = 80`: Ancho del mapa en caracteres.
- `MAP_HEIGHT = 25`: Altura del mapa.
- `MIN_ROOM_SIZE = 5`: Tamaño mínimo de cada sala generada.

Se exploró experimentalmente el impacto de modificar estos parámetros. Por ejemplo, valores menores a 5 para `MIN_ROOM_SIZE` generaban salas muy pequeñas que reducían la legibilidad.

El flujo del algoritmo fue:

1. Crear el nodo raíz con el `Room` principal.
2. Aplicar `splitNode()` recursivamente para dividir el espacio.
3. Generar habitaciones dentro de los nodos hoja.
4. Conectar las habitaciones secuencialmente con pasillos.
5. Renderizar el mapa en consola.

4. Resultados

Se presentan mapas generados mediante ejecuciones independientes del programa. A continuación, se muestran capturas de los resultados con diferentes configuraciones:

Figura 1: Mapa generado con `MAP_WIDTH = 80`, `MAP_HEIGHT = 25`, `MIN_ROOM_SIZE = 5`

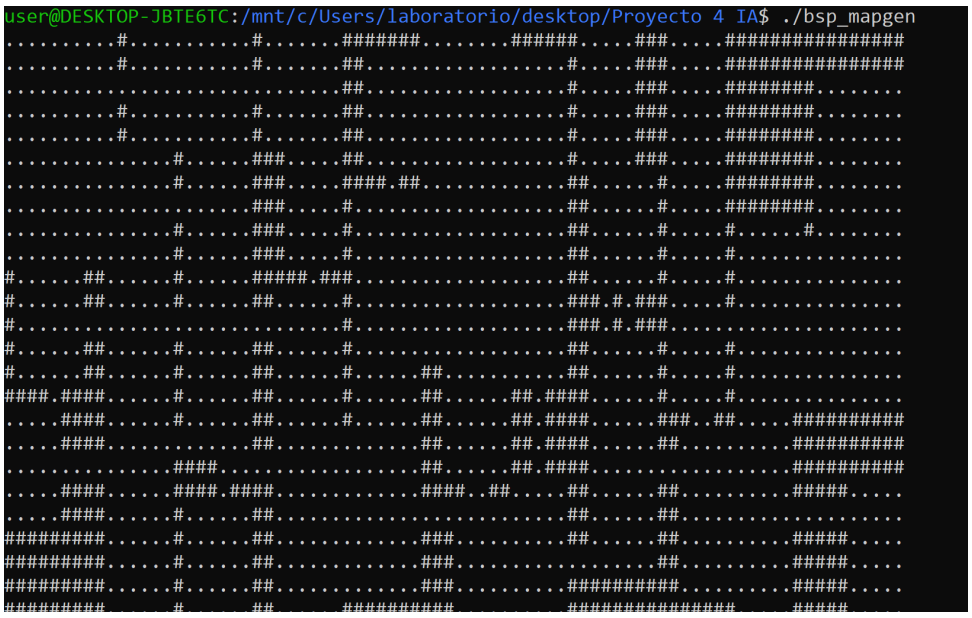
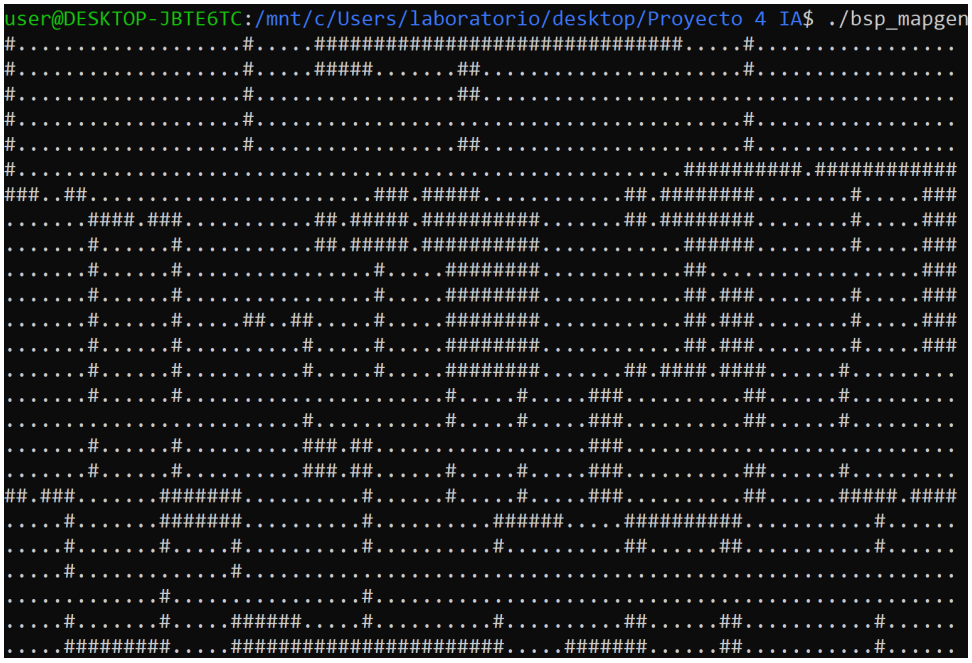


Figura 2: Variación con `MIN_ROOM_SIZE = 8`, resultando en menos habitaciones pero más grandes



5. Discusión

El algoritmo BSP permite generar mapas funcionales con una estructura jerárquica clara. Su principal ventaja es que cada ejecución genera una estructura distinta pero coherente.

Se detectó que usar `MIN_ROOM_SIZE` muy bajo incrementa el número de habitaciones, pero muchas de ellas se superponen o resultan poco visibles. Al aumentar ese valor, el mapa se vuelve más legible, aunque menos denso.

6. Conclusiones

La técnica de *Binary Space Partitioning* es adecuada para la generación de mapas como es en el caso de Hades. Su implementación en C++ resulta sencilla y eficiente, esto a pesar del esfuerzo en comprender la estructura base de este. Los resultados muestran mapas variados, replicables y con estructura jerárquica clara, tambien mencionar que al comienzo me costo bastante entender la logica dentro de las opciones que tenia estaba the binding of isaac, dead cells y darkwood, pero me intrigaba bastante saber como funcionaba en hades dado el sistema de salas que este tiene cuyo orden siempre va variando.

Para finalizar en este trabajo se demuestra que técnicas de división espacial como BSP pueden adaptarse fácilmente al diseño de niveles, siendo una opción viable para proyectos con estética *rogue-like* y objetivos de rejugabilidad.

7. Referencias

- Metodo del arbol BSP . https://www.youtube.com/watch?v=AAo9-8M3oIApp=ugMGCgJlcxABu
- BSP-Dungeon-Generation-Study https://github.com/EduardoLopes/BSP-Dungeon-Generation-Study
- Wikipedia: Binary Space Partitioning. https://en.wikipedia.org/wiki/Binary$_s pace_p artitioning Ay$
- Literatura encontrada https://arxiv.org/abs/2410.15644