Generación Procedural

Caballero Jiménez, Oscar Emilio

27 de mayo de 2025

1. Introducción

La generación procedural es una técnica utilizada para crear datos (como imágenes, sonidos, niveles de videojuegos, mundos virtuales, etc.) de forma automática mediante algoritmos y reglas predefinidas, en lugar de hacerlo manualmente.

En IA, la generación procedural adquiere un matiz más avanzado al combinarse con técnicas como redes neuronales y aprendizaje automático (ML). Aquí el sistema no solo sigue reglas fijas, sino que aprende patrones para generar contenido nuevo.

1.1. Características clave

- **Aleatoriedad controlada**: Usa semillas (seeds) para generar resultados únicos pero reproducibles.
- **Eficiencia**: Permite crear contenido complejo sin almacenarlo todo previamente (ejemplo, mundos como en Minecraft).
- **Escalabilidad**: Ideal para sistemas donde el contenido manual sería muy poco viable o difícil de hacer (ejemplo, galaxias en No Man's Sky).

1.2. Ejemplos clásicos de uso

- Videojuegos: Mapas, mazmorras, texturas o vegetación.
- Gráficos 3D: Terrenos, nubes o efectos de fuego/agua simulados.
- Música: Composiciones algorítmicas basadas en reglas armónicas.

1.3. Ejemplos enfocados en IA

- Redes Generativas Adversariales (GANs): Dos redes neuronales compiten (una genera contenido y otra lo evalúa), produciendo resultados hiperrealistas.

- Autocodificadores variacionales (VAEs): Generan datos similares a los de entrenamiento (ejemplo, diseños de muebles o arte abstracto).
- Modelos de lenguaje (como GPT): Generan texto coherente a partir de prompts (peticiones), simulando creatividad humana.

2. Desarrollo

Usaremos a Borderlands como nuestro juego ejemplo para el uso de generación procedural. Borderlands cuenta con un método de PCG (Procedural Content Generation), llamado Runtime Random Level Generation, el cual se encarga de ir generando el nivel mientras el juego es jugado, ofreciendo así una infinidad de modos de jugar el mismo nivel.

2.1. Generación de terreno

Para la generación de terreno podemos usarsemillas s(seeds), pero me gustaría enfocarme en el método Runtime Random Level Generation que usé de ejemplo para Borderlands. Este método se puede hacer de muchas maneras, pero una de ellas es el algoritmo Diamond-Square o Diamond-Square Algorithm, el cual es un método para generar heightmaps (una imagen rasterizada usada principalmente como Discrete Global Grid en modelado de elevación secundario) para gráficos por computadora.

2.1.1. Descripción del algoritmo

Empieza con una matriz cuadrada con width y height de $2^n + 1$. Las cuatro esquinas de la matriz se asignan con valores iniciales. Los pasos The Diamond y square se realizan alternadamente hasta que todos los valores de la matriz sean llenados.

- **The diamond step:** Para cada cuadrado de la matriz, establezca el punto medio de ese cuadrado como el promedio de los cuatro puntos de las esquinas más un valor aleatorio.
- **The square step:::** Para cada diamante de la matriz, establezca el punto medio de ese diamante como el promedio de los cuatro puntos de las esquinas más un valor aleatorio.

Cada valor aleatorio se multiplica por una constante de escala, que disminuye con cada iteración por un factor de 2^{-h}, donde h es un valor entre 0.0 y 1.0 (los valores más bajos producen un terreno más accidentado).

2.2. Implementación en un videojuego

La incorporación a mi videojuego supondría un reto debido a que debería cambiar la perspectiva del juego (ya que se ve desde arriba para poder ver

la interacción de IA's), pero para la incorporación se tendría que cambiar la cámara a primera persona; las IA's se tendrían que ir generando con el terreno y mantenerse, lo cual implicaría, conforme el juego avance, una mayor carga computacional y de gráficos.

La experiencia de juego sería mucho más satisfactoria dado que el terreno cambiaría siempre en cada partida jugada y no habría forma de memorizarse el terreno, lo cual siempre traería una nueva experiencia.

3. Conclusiones

3.1. Ventajas de la generación procedural

- **Eficiencia en recursos**
  - Ahorro de espacio: No requiere almacenar grandes cantidades de datos.
  - Escalabilidad: Puede crear mundos o contenidos infinitos sin necesidad de diseño manual.
- **Variedad y diversidad**
  - Contenido único: Cada ejecución puede producir resultados distintos.
  - Adaptabilidad: Se ajusta dinámicamente (cambian según la habilidad del jugador).
- **Reducción de costos**
  - Menor trabajo manual: Ideal para estudios pequeños o proyectos con limitaciones de presupuesto.
- **Aplicaciones en IA**
  - Entrenamiento de modelos: Genera datos sintéticos para entrenar redes neuronales.
  - Creatividad asistida: Herramientas como DALL-E o MidJourney permiten crear arte rápido basado en reglas aprendidas.

3.2. Desventajas de la Generación Procedural

- **Falta de control fino**
  - Resultados impredecibles: Pueden generarse elementos incoherentes o no deseados.
  - Dificultad para narrativa: En videojuegos, es complejo generar historias con sentido sin intervención humana.
- **Calidad inconsistente**

    - Repetición de patrones: Si los algoritmos no son sofisticados, el contenido puede volverse repetitivo.

- **Curva de aprendizaje**
    - Complejidad técnica: Diseñar algoritmos efectivos requiere conocimientos avanzados en matemáticas y programación.

- **Limitaciones en IA**
    - Sesgos en datos: Si el modelo se entrena con datos sesgados, generará contenido sesgado.
    - Falta de .ºriginalidad real": La IA combina patrones existentes, pero no tiene creatividad humana genuina.

En videojuegos la generación procedural ha dado un step up en términos de experiencia de juego, dando como opción una infinidad de formas de disfrutar el mismo juego.

4. Referencias

- Libro: Procedural Generation in Game Design (Tanya Short, Tarn Adams, 2017). Cubre fundamentos y aplicaciones en videojuegos.
- Artículo: Procedural Content Generation: A Taxonomy and Review (Togelius et al., 2011). Clasificación académica de técnicas de generación procedural. (`https://ieeexplore.ieee.org/document/5756645`)
- Recurso: Procedural Generation Wiki. Ejemplos prácticos y algoritmos. (`http://pcg.wikidot.com/`)
- Recurso Borderlands: PCG Types en Borderlands. (`http://pcg.wikidot.com/pcg-games:borderlands`)
- Generative Adversarial Networks (Goodfellow et al., 2014) — Paper fundacional. (`https://arxiv.org/abs/1406.2661`)
- StyleGAN (Karras et al., 2019) mejoras en generación de imágenes. (`https://arxiv.org/abs/1812.04948`)
- Auto-Encoding Variational Bayes (Kingma & Welling, 2013). (`https://arxiv.org/abs/1312.6114`)
- Language Models are Few-Shot Learners (Brown et al., 2020) — Paper de GPT-3. (`https://arxiv.org/abs/2005.14165`)
- Recurso: Diamond-square algorithm. (`https://en.wikipedia.org/wiki/Diamond-square_algorithm`)