Implementación de algoritmos de generación procedural para la creación de niveles de Geometry Dash

Simon Gallardo
sgallardo23@alumnos.utalca.cl

Rodrigo Díaz
rdiaz23@alumnos.utalca.cl

Junio 2025

1 Abstract

La generación procedural de contenido puede ser aplicada en muchos contextos, entre ellos los videojuegos, se hizo un experimento con el juego geomety dash con el objetivo de saber si es realmente viable crear niveles jugables con este metodo. Para la aplicación del estudio se usó el algoritmo de las cadenas de Markov el cual permite crear contenido en base fragmentos basados en parte de ese contenido que se tiene certeza de que funciona. Se crearon 3 niveles en 3 dificultades diferentes dando como resultado que si es posible crear niveles para el juego mediante este metodo y puede ser escalable para así hacer niveles incluso más fieles a lo que se vería en el juego oficial.

2 Introducción

Geometry dash es un juego de plataformas que salió el año 2013 por RobTop games en el que el objetivo principal es terminal niveles sorteando obstaculos como pinchos, bloques, cierras, etc. Este juego se diferencia del resto de su genero ya que permite una gran cantidad de vehiculos los cuales se controlan de distintas formas lo que lo hace bastante retador aveces. Debido a como se estructuran estas etapas creemos que este juego puede ser victima del estudio para generación procedural de contenido mediante el algoritmo de cadenas de Markov ya que permite crear en base a patrones ya establecidos, Como explica Barriola and Dotta, 2016 las cadenas de Markov se basan en matrices de trancisión donde cada fila contiene las probabilidadesde pasar de un estado a otro dando como resultado comportamientos secuenciales, donde la sección A puede ir a B y luego a C, por ende A también puede ir a C directamente Gagniuc, 2017. Dado ese comportamiento creemos que el resultado para generar niveles de geometry dash puede ser interesante.

3 Videojuegos y Cadenas de Markov (estado del arte)

Las cadenas de Markov son un modelo matematico e implementable como algoritmo que describe sistemas de transición entre estados con cierta probabilidad. Este se usa para distintos propositos relacionados con la predicción de comportamientos, probabilidades, y en los que nos enfocaremos en esta investigación, la generación de contido proceduralmente. En videojuegos particularmente se han usado para la predicción de movimiento en juegos del genero MOBA Zhang et al., 2019.Y el caso que nos incumbe, la generación de niveles y mapas, bien funcionales y especialmente probados y utilizados en videojuegos 2D de plataformas como Super Mario Bros o Kid IcarusSnodgrass and Ontañón, 2017.

4 Metodologia

Para poner a prueba este experimento lo que haremos será crear distintas planillas o slices que representarán distintos elementos del juego. (ya que se trata de motivos de estudio lo más complejo que habrá será el cambio de cubo a nave y viceversa). Se crearán 3 maquetas distintas basadas en los slices que se usaron para "entrenar" al algoritmo y posteriormente las pasaremos al juego mediante su editor de niveles y evaluaremos los siguientes criterios:

- Completabilidad
- Variedad y jugabilidad

4.1 Parametros

Los parametros que considera el algoritmo son principalmente 2 los cuales son la dificultad del nivel y la duración de este. La dificultad será medida en 3 posibles variantes:

- Facil
- Medio
- Dificil

y la forma en la que afectará a la creación del nivel es que dependiendo de la dificultad el algoritmo premia o penaliza ciertos elementos en la creación del nivel como los pinchos por ejemplo, en dificultad facil se trataran de evitar, mierntras que en dificil se premia la aparición de estos. La duración contempla cuantos slices considera al momento de crear el nivel, por lo que mientras mayor sea el numero, más slices considerará dando como resultado un nivel largo o corto en función de ese parametro. Por ultimo otro parametro que afecta a la creación de un nivel es la matriz de adyacencia que maneja las probabilidades de aparición de los slices en base a desabilidad, dificultad y trancisiones lógicas.

4.2 Simbologia

- `*` = pincho
- `#` = Bloque
- `C` = Portal de cubo
- `S` = Portal de nave
- `_` = suelo

como ejemplo un nivel podria verse de la siguente manera:

```
#### C ####
|       *  #
|          #
#### S ####
```

5 Analisis de resultado

1 Facil En las figuras de la 1 a la 4 podemos ver el nivel generado para la dificultad facil, podemos observar la completa aucencia de pinchos y predominación de bloques por lo que en teoria el algoritmo funciona como esperabamos para esta dificultad, jugablemente el nivel es competable, pero peca de ser bastante repetitivo al haber muchas veces repetido el patrón de la nave, esto tambien da como resultado que en ocaciones terminemos jugando con la nave secciones que estaban pensadas para el cubo, pero nunca se generó algo que nos hiciera jugar con el cubo una sección de nave, por lo que el algoritmo funciona bien haciendo uso de las probabilidades de uso para las slices priorizando secciones jugables.

2 Medio En las figuras de la 5 a la 9 encontramos el nivel generado en medio, el resultado es similar al primero con la diferencia que ahora hay secciones que contiene pinchos, pero tiene un salto demasiado preciso lo que permite concluir que por la forma en la que estan hechos los slices con los que se entrenó la IA existe una posibilidad de que los niveles de otras dificultades sean más dificil que esa propia dificultad

3 Dificil Finalmente en la dificultad más alta podemos ver que los pinchos salen con más frecuencia que las otras 2 y en 2 ocaciones hay una seccion con la nave bastante estrecha por lo que se cumple con la dificultad.

5.1 Niveles generados

Facil:

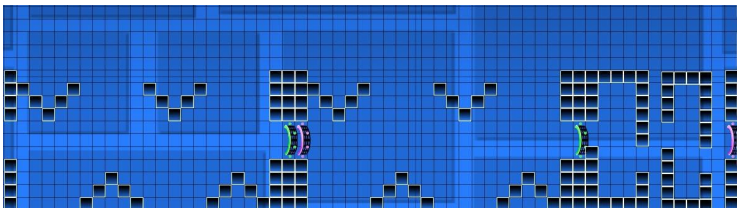

Figure 1: Nivel facil parte 1.

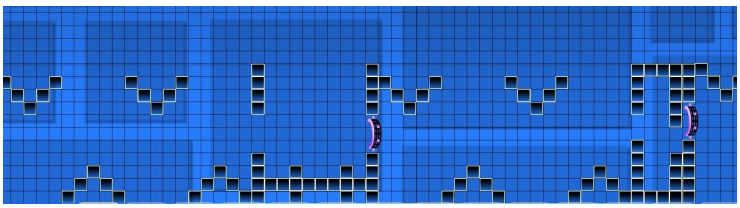

Figure 2: Nivel facil parte 2.

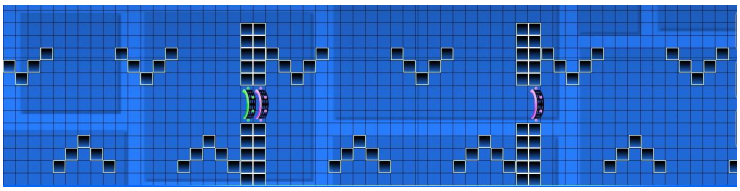

Figure 3: Nivel facil parte 3.

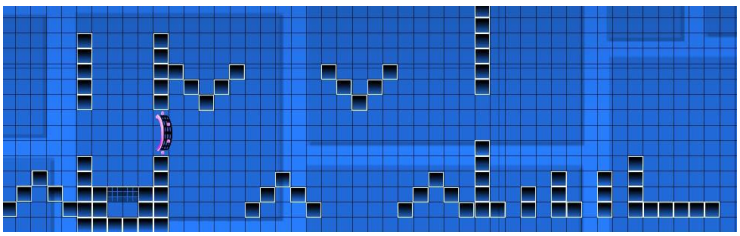

Figure 4: Nivel facil parte 4.

Medio:

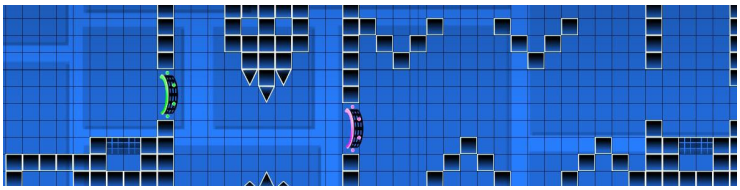

Figure 5: Nivel medio parte 1.

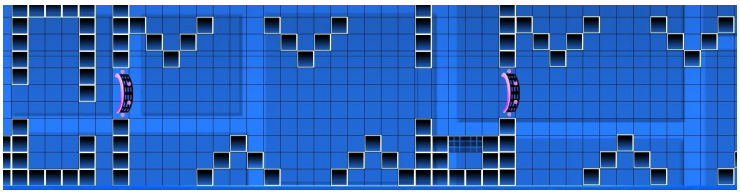

Figure 6: Nivel medio parte 2.

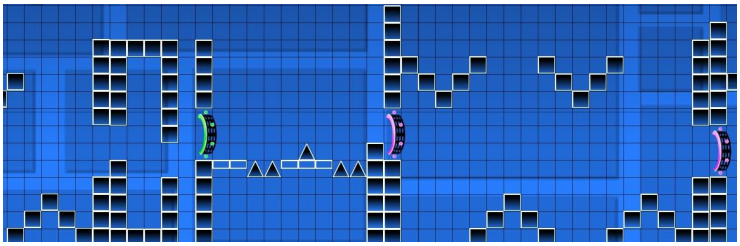

Figure 7: Nivel medio parte 3.

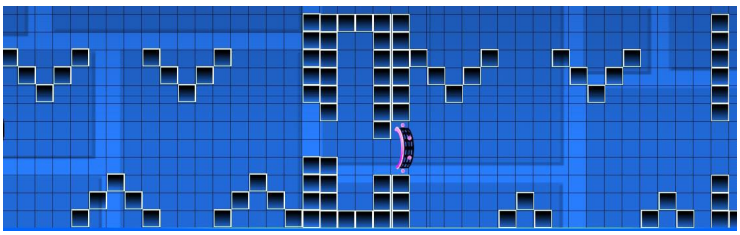

Figure 8: Nivel medio parte 4.

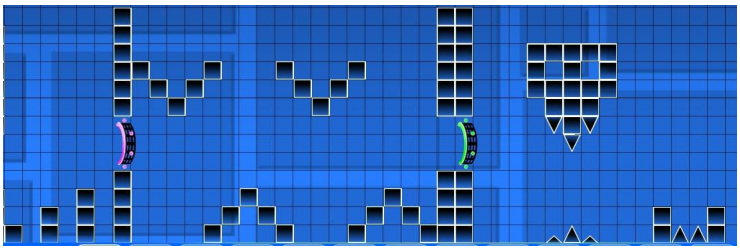

Figure 9: Nivel medio parte 5.

Dificil:

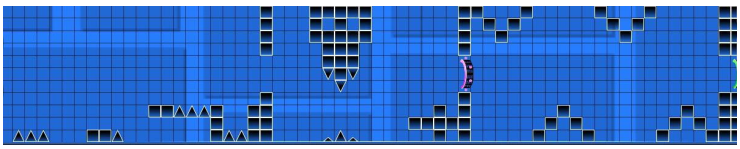

Figure 10: Nivel dificil parte 1.

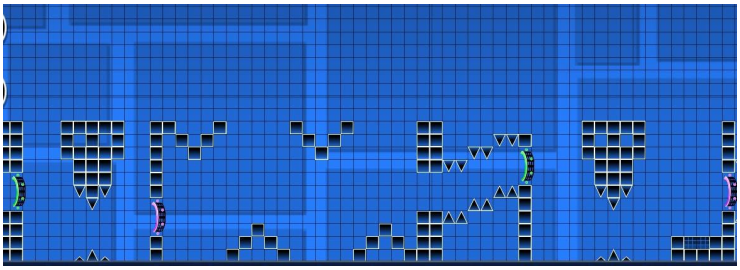

Figure 11: Nivel dificil parte 2.

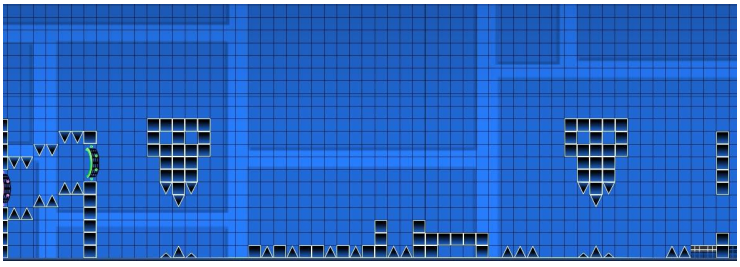

Figure 12: Nivel dificil parte 3.

5.2 Notas*

- Las maquetas generadas por el algoritmo serán adjuntadas en un archivo txt
- ID niveles
  - Facil: 121681697
  - Medio: 121681692
  - Dificil: 1216811701

6 Discusión

Llegados a este punto considerando como funciono el algoritmo nos queda hacernos la pregunta ¿funcionó como esperabamos? la respuesta es que si, ya que como vimos el algoritmo fue capaz de generar niveles completamente jugables, sumado a eso se podrian cambiar ciertos parametros como hacer cada slice más detallado en lugar de hacer secciones completas y se podria aumentar aún más la calidad del nivel generado, del mismo modo podriamos considerar otros vehiculos como el Ufo o la Wave e incluso agregar portales como los gravitatorios o las orbes para hacer niveles aún más completo. Como podriamos hacer esto, modificando la matriz de adyacencia para que contemple tambien las nuevas funciones y considere probabilidades basadas en estos nuevos parametros, asi mismo podriamos incluir más dificultades como la demon que podria priorizar espacios estrechos o de incluirlos, portales gravitatorios mezclados con algún vehiculo como la nave.

7 Conclusiones

En base a estos resultados podemos ver que es posible generar niveles funcionales para geometry dash basado en el algoritmo de cadenas de markov y que lo repetitivo de los niveles generados se debe a la falta de contenido con el que el algoritmo fue entrenado, pero si hicieramos que cada slice fuera más detallado en lugar de ser secciones completas podriamos tener niveles que se asemejen más a los diseñados por su creador, tambien se podria entrenar al algoritmo para que cambie el como considera las probabilidades y evitar casos como el del salto preciso de la dificultad media, pero sin duda es una herramienta funcional, por lo mismo mejorarla para que considere otras funciones del juego como vehiculos, portales, nuevas dificultades o incluso cambios de velocidad podrian hacer que se generacen niveles mucho más interesantes de los presentados.

References

Barriola, J. M., & Dotta, M. (2016). ¿cómo funciona google? el algoritmo pagerank, diagramas de grafos y cadenas de markov. *Revista de Investigación en Modelos Matemáticos Aplicados a la Gestión y la Economía*, *3*(3), 9–30.

Gagniuc, P. A. (2017). *Markov chains: From theory to implementation and experimentation*. John Wiley & Sons. https://doi.org/10.1002/9781119387596

Snodgrass, S., & Ontañón, S. (2017). Learning to generate video game maps using markov models. *IEEE Transactions on Computational Intelligence and AI in Games*, *9*, 410–422. https://doi.org/10.1109/TCIAIG.2016.2623560

Zhang, Q., Tan, Z., & Xu, Y. (2019). The application of markov chains in multiplayer online battle arena (moba) games. In *Irc-set 2018*. Springer. https://doi.org/10.1007/978-981-32-9828-6_9