Testing wave function collapse to represent behaviors in video games

Adrián Ramos [1], Micaela Y. Martín [1], Miguel Chover [1]

[1] *Universitat Jaume I, Adv. Vicent Sos Baynat s/n, Castellón, 12071, Spain*

Abstract

One avenue of research with great viability for the development of video games is that of methods that allow the generation of procedural content. In recent years, the use of the Wave Function Collapse algorithm to create this type of content stands out. Although the algorithm has been used mainly for texture generation and level creation, its application to other fields of video games has been less explored and has a higher margin of application. The work developed proposes its use to procedurally generate the behavior of the Non-Player Characters of a video game. The solution presented is based on the creation of behavioral structures such as decision trees. As an example of how the algorithm works, a simple video game has been created.

Keywords

Wave Function Collapse, Artificial Intelligence, Procedural Content, Non-Player Characters, Video Game.

1. Introducción

En relación con las técnicas de creación procedimental, en los últimos años destaca la utilización del algoritmo de Colapso de la Función de Onda (Wave Function Collapse - WFC) [1]. Este algoritmo se ha utilizado con éxito en diferentes campos como en el de generación de texturas [2], programación con restricciones [3] [4] y el diseño y modelado [5] [6], entre otros.

En concreto, el trabajo propuesto utiliza el algoritmo, para generar procedimentalmente el comportamiento de los personajes no jugadores (*Non-Player Characters* - NPC) de un videojuego. La solución que se presenta está basada en la creación de estructuras formadas por bloques que representan diferentes comportamientos de los NPCs. Los árboles de decisión son construidos y organizados usando el algoritmo de WFC que, mediante las restricciones, consigue dotar de sentido a las estructuras que crea, sin perder su naturaleza aleatoria. Por lo tanto, esto nos permite generar una gama de distintos comportamientos a partir de la generación simultánea de diversos árboles de decisión que satisfacen las restricciones impuestas.

2. Bases del algoritmo de Colapso de la Función de Onda

Los elementos básicos del algoritmo WFC son las variables, el dominio y las restricciones. Las variables son todos los elementos del problema a descubrir. En un sudoku, por ejemplo, serían cada una de las casillas que forman el tablero. A su vez, cada variable cuenta con su propio dominio, representado por los valores que puede adquirir. En el caso del sudoku, el dominio serían los números del uno al nueve. Por último, las restricciones, son las reglas que se establecen para modificar el comportamiento del algoritmo y adaptar el contenido generado a las características previstas. En el

I Congreso Español de Videojuegos, December 1–2, 2022, Madrid, Spain

EMAIL: al385729@uji.es (A. Ramos); micmarti@uji.es (M. Y. Martín); chover@uji.es (M. Chover)

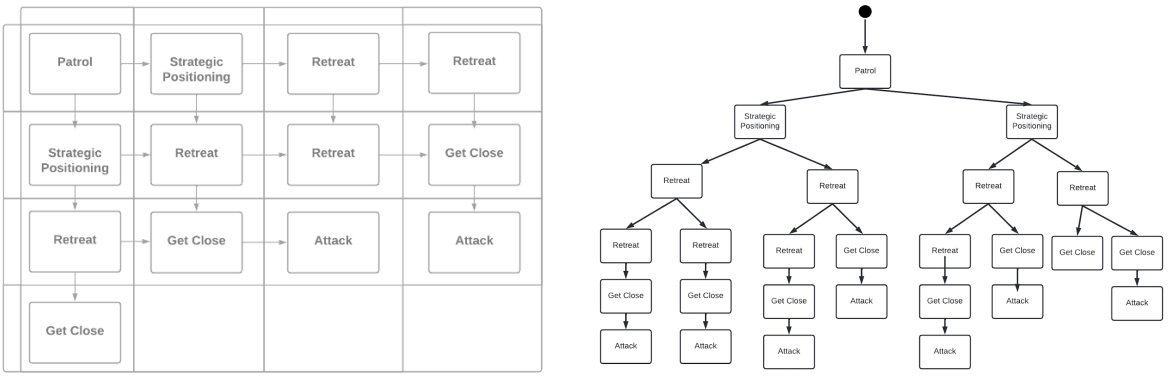
ORCID: 0000-0001-9341-8331 (A. Ramos); 0000-0001-6100-7538 (M. Y. Martín); 0000-0002-0525-7038 (M. Chover)

ejemplo del sudoku implicaría que no se repitiera el mismo número en una fila, en una columna ni en un cuadrante de este.

3. Aplicación del algoritmo a la generación de comportamiento

El algoritmo propuesto se basa en los mismos principios que para la creación de teselas utilizando WFC [6] pero en este caso se utilizan bloques de comportamiento. Estos bloques son contenedores que incluyen el código necesario para expresar el comportamiento de los NPCs. Cada uno de los bloques de comportamiento tiene implícitas las condiciones que provocan la mutación del comportamiento externo y solo se ejecutan cuando estas se cumplen, así conseguimos que el flujo de ejecución vaya de la raíz a las hojas del árbol.

El algoritmo se ha diseñado para que obtenga como resultado una matriz de comportamientos aleatorios (de forma similar a como se generan imágenes de teselas [6]). Esta matriz, se traduce fácilmente en un árbol de decisión (ver figura 1) [7].



(a) (b)

Figura 1: Matriz de comportamientos (a) y árbol de decisión del comportamiento del NPC (b).

4. Videojuego desarrollado

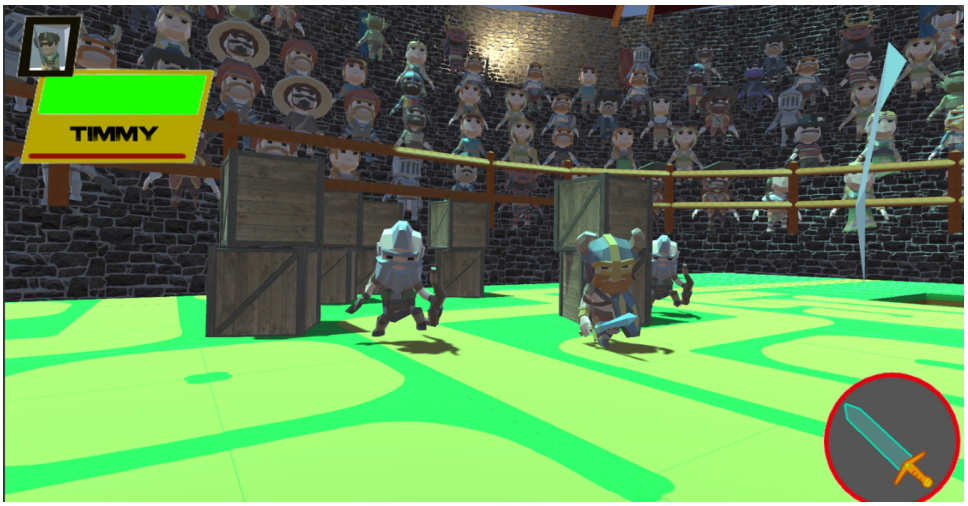


Figura 2: NPCs en el videojuego desarrollado.

El algoritmo se ha aplicado con éxito para generar comportamientos aleatorios en NPCs de un videojuego (ver figura 2). El juego tiene dos tipos de enemigos con los que hay que luchar: los

arqueros y los espadachines. Cada uno de ellos, tiene cinco posibles comportamientos, tres son comunes y dos específicos:

Los comportamientos generales son:

- Patrulla. Movimiento de ida y vuelta entre dos puntos.
- Retirada. Movimiento de vuelta a la base.
- Posicionamiento estratégico. Movimiento hacia un lugar estratégico.

Los comportamientos específicos de los arqueros:

- Alejarse de un objetivo. Movimiento en sentido contrario al objetivo.
- Disparar a distancia. Disparo al jugador a partir de una determinada distancia.

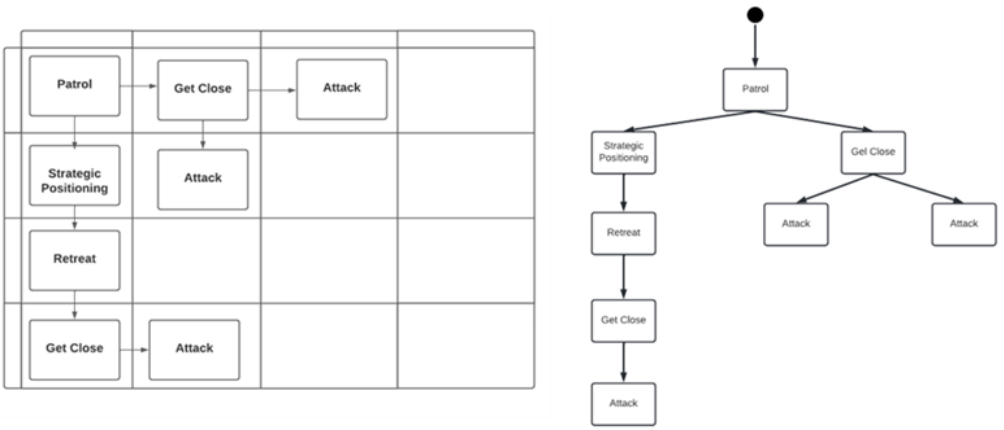
Los comportamientos específicos de los espadachines:

- Acercarse a un objetivo. Movimiento hacia uno de los objetivos.
- Ataque cercano. Atacar al jugador a corta distancia.

Tanto el jugador como los enemigos, se incluyen en una arena donde se produce el combate. Lo interesante del sistema es que permite ver comportamientos diferentes para cada uno de los enemigos ya que sus acciones han sido generadas de forma procedimental, evitando los comportamientos repetitivos y permitiendo que estos sean diferentes para cada partida.

5. Resultados

Como ejemplo de los resultados obtenidos se presentan las matrices de comportamientos y los árboles de decisión asociados, para un NPC de tipo espadachín. De esta forma, se puede ver cómo el espadachín 1 puede comportarse con el árbol de decisión de la figura 1 y el espadachín 2 con el árbol de decisión de la figura 3.



(a) (b)

Figura 3: Matriz de comportamientos (a) y árbol de decisión (b) del espadachín 2.

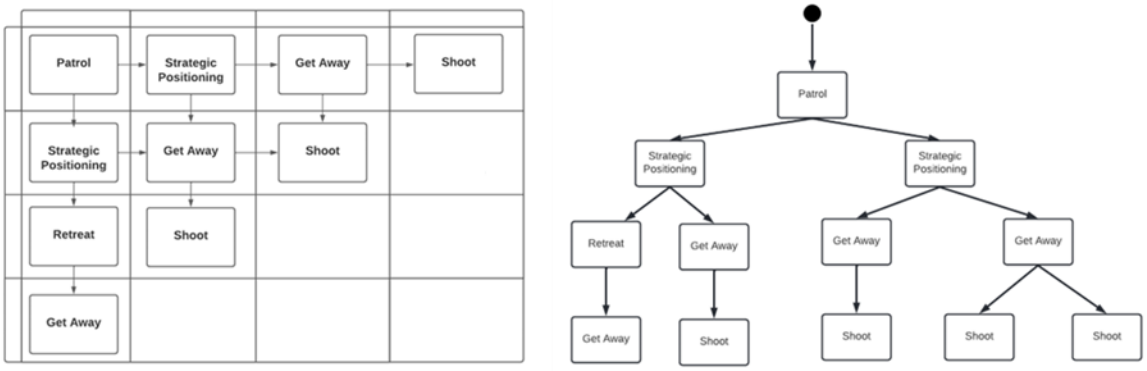
Un ejemplo de la generación aleatoria del comportamiento para el NPC tipo arquero se puede ver en la figura 4.

La estructura de estos árboles está totalmente determinada por las restricciones que se han establecido para combinar los bloques de comportamiento, por el tamaño máximo de la matriz que se ha utilizado como base y por el número de hijos máximo que se asignan a cada bloque de comportamiento. Estas variables pueden adaptarse a las necesidades del juego en el que queramos generar el contenido, afectando sólo el tiempo de procesamiento.

5. Conclusiones

Como se ha descrito en este trabajo, el uso del algoritmo WFC se está extendiendo por la industria a una gran velocidad y pueden encontrarse muchos proyectos que lo utilizan para la generación de

niveles [8] y el modelado geométrico [9]. En este trabajo se ha utilizado la misma idea para generar comportamientos aleatorios en NPCs, generando una matriz de comportamientos que posteriormente se transforma en un árbol de decisión. Los resultados obtenidos muestran el gran potencial de la propuesta para la definición de comportamientos en videojuegos.



(a) (b)

Figura 4: Matriz de comportamientos (a) y árbol de decisión (b) del arquero.

6. Agradecimientos

Este trabajo ha sido posible gracias al programa "Estudia e investiga en la UJI" de la Universitat Jaume I y a los proyectos de investigación: PID2019-106426RB-C32 financiado por MCIN/AEI/ 10.13039 / 501100011033 y FEDER "Una manera de hacer Europa", PDC2021-120997-C31 financiado por MCIN/AEI/10.13039/501100011033 y la Unión Europea "NextGenerationEU"/PRTR, y CIAICO/2021/037 financiado por Generalitat Valenciana.

7. Referencias bibliográficas

[1] Isaac Karth and Adam M. Smith. 2017. WaveFunctionCollapse is Constraint Solving in the Wild. In Proceedings of FDG'17, Hyannis, MA, USA, August 14-17, 2017, 10 pages. https://doi.org/10.1145/3102071.3110566.

[2] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. 2001. Real-time texture synthesis by patch-based sampling. ACM Transactions on Graphics (ToG) 20, 3 (2001), 127–150. https://doi.org/10.1145/501786.501787.

[3] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. 2012. Conflict-driven answer set solving: From theory to practice. Artificial Intelligence 187 (2012), 52–89. https://doi.org/10.1016/j.artint.2012.04.001.

[4] G. Smith, J. Whitehead and M. Mateas, "Tanagra: Reactive Planning and Constraint Solving for Mixed-Initiative Level Design," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 3, no. 3, pp. 201-215, Sept. 2011, doi: 10.1109/TCIAIG.2011.2159716.

[5] Maxim Gumin. 2016. WaveFunctionCollapse. https://github.com/mxgmn/ WaveFunctionCollapse. GitHub repository (2016). https://github.com/mxgmn/WaveFunctionCollapse

[6] Maxim Gumin. 2017. WaveFunctionCollapse Readme.md. (18 May 2017). Retrieved May 20, 2017, from https://github.com/mxgmn/WaveFunctionCollapse/blob/master/README.md

[7] J. Millington, I. & Funge. Artificial Intelligence for Games. Taylor & Francis, 2009.

[8] Alexander Gellel and Penny Sweetser. 2020. A Hybrid Approach to Procedural Generation of Roguelike Video Game Levels. In the International Conference on the Foundations of Digital Games (FDG '20). Association for Computing Machinery, New York, NY, USA, Article 3, 1–10. https://doi.org/10.1145/3402942.3402945

[9] R. van der Linden, R. Lopes and R. Bidarra, "Procedural Generation of Dungeons," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 6, no. 1, pp. 78-89, March 2014, doi: 10.1109/TCIAIG.2013.2290371.