Khaldun: GOAP for both Procedural Level generation and NPC Behaviors

Mael Ahmad Addoum*, Jannah Mekhaemar*, Maxime Rouffet*, Éric Jacopin†
* International School of Video Game and Animation 3D-FX, *Isart Digital*, Paris, France
†CREC, *Écoles de Saint-Cyr Coëtquidan*, Guer, France

***Abstract*—We investigate the capability of Goal Oriented Action Planning (GOAP) to perform a double-objective: (1) to control the Non-Player Character (NPC) behaviors, and (2) to generate dungeon levels. We demonstrate its application for a 2D custom platformer roguelike game. Our first results show GOAP's ability to generate a wide range of playable and stable dungeon levels while producing unpredictable NPC actions.**

***Index Terms*—GOAP, Procedural Level Generation.**

I. Introduction

We address the use of Game AI Planning for online level generation. Automatically generating levels is a challenge and a creatively demanding task where both functional and structural requirements must be met [1]. Related works mainly concerning the search-based methods [2], while Hauck and Aranha proposed a new graph-based grammars system for *Super Mario Bros* [1]. Khalifa et al. combine evolutionary algorithms with quality-diversity algorithms that create levels with varying characteristics [3]. Du et al. have coupled both search and genetic algorithms to generate respectively the player path and monster sequences that make the levels playable [4]. GOAP has been mainly used for generating NPC behaviors. In our custom platformer game named *khaldun*, we investigate the use of GOAP to generate dungeon levels in addition to NPC behaviors. *Khaldun*, Fig. 1, https://victorcavagnac.itch.io/khaldun, is a 2D roguelike game where the player incarnates a lone traveler seeking the Fountain of Eternity to cure his illness. To the best of our knowledge, no other study has reported the use of GOAP to automatically generate platformer levels, with one exception of puzzle generation in an adventure game [5].

II. Procedural Level Generation

The first step consists in hand-designing rectangular rooms where the fixed elements and enemies are carefully placed within a 32×19 map. The designer can easily sketch the desired structure in a very simple way (Fig. 2-top). The list of these elements with their colors and layout are given in Table I .

After adding decorations, the designed rooms are then generated and transformed as prefabs assets in the game engine (Fig. 2-bottom). Rooms are equivalent to actions that are used in the behavioral-planning algorithm while the level is equivalent to its plan. Each generated room has a list of preconditions and post-conditions that should be satisfied in order to place it in a suitable position within the level. These conditions are represented by the entrances and exits on the left, right, top or bottom of the room. It is important to sequence the rooms that will hold game events so as to guide the player progression. In *Kaldhun*, the planner locates the rooms with respect to randomly weighted coefficients taking into account the connection between the neighbored rooms. These coefficients aim at adding variety to the dungeon and to place top and bottom strategics rooms in order to avoid simply aligned rooms. The constructed plan is therefore a sequence of several connected rooms that constitute the entire dungeon level where at least one non linear valid path is ensured.

978-1-6654-3886-5/21/$31.00 ©2021 IEEE

Fig. 1. Three enemies chasing the player in *Kaldhun*.

III. Results

Based on designers' parameters, a generated level aims to last between 10-15min of gameplay. Figure 3 depicts two different examples of procedurally generated levels; both levels are visually appealing and pleasing and guarantee at least one solvable path traveling from left to right. The player can explore the dungeon in the vertical direction due to the top and bottom rooms' positioning. This is ensured thanks to the random coefficients used in the generator that avoid obtaining a straightforward boring path. The computation cost to produce a whole level is neglected allowing hence an online generation. Empirical tests showed that the generated dungeons are unique, solvable, and have high replayability while respecting game design. Moreover, our algorithm leads to unpredictable NPC actions which avoids redundant behaviors. Despite the simultaneous plan generation for the NPC present in a single room, as in the case of figure 1, the

TABLE I
MAIN ELEMENTS USED IN THE PROCEDURAL LEVEL GENERATION.

Elements	Color	Layout
Floor		
Wall		
Rope		
Platform		
Chest		
Spawn location for the enemy		
Teleport Stele		

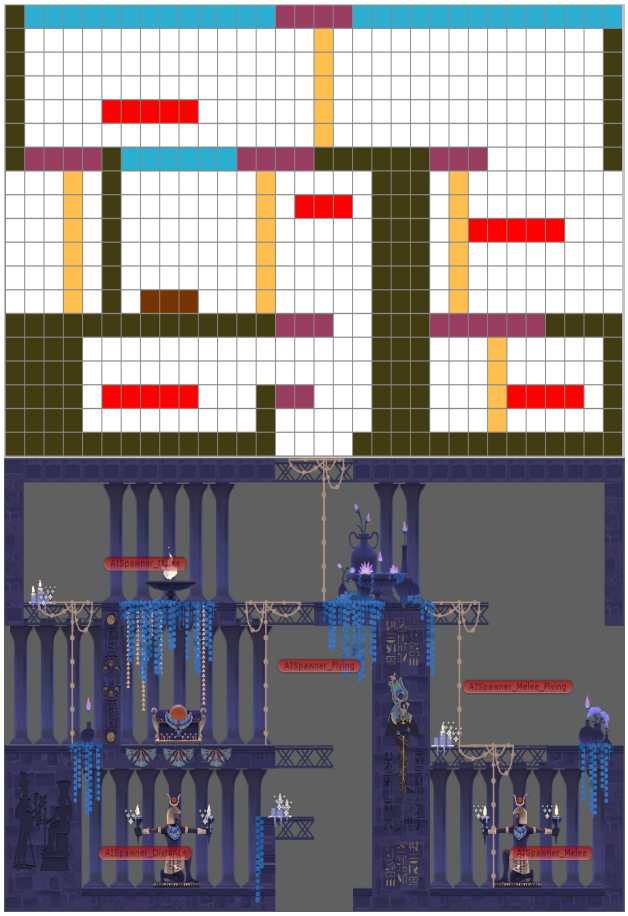

Fig. 2. Hand-designed 32 × 19 sketch map (top), and the corresponding generated room (bottom).

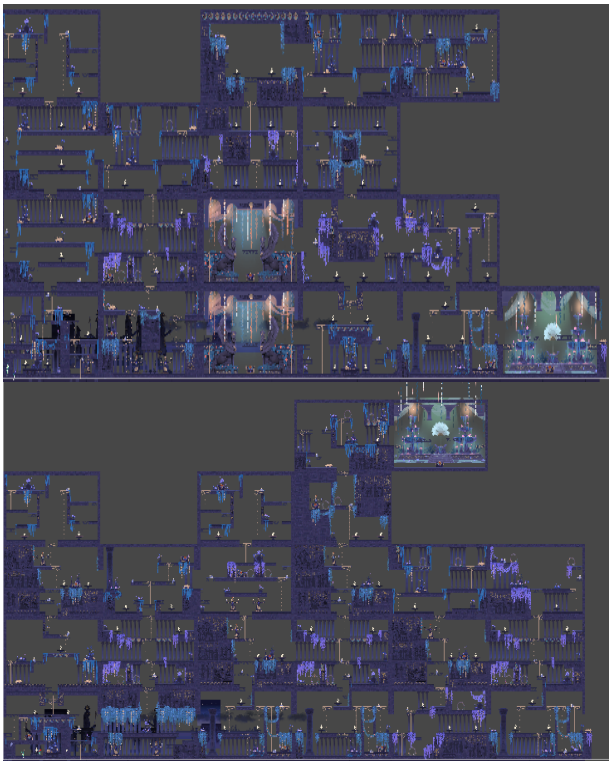

Fig. 3. Two examples of dungeons levels automatically generated with GOAP.

planning runtime remains efficient and does not impact the game quality. Finally, designers can easily add or remove actions or room structures within the planner so as to evolve the developed game and extend its durability.

IV. CONCLUSIONS

We presented a GOAP architecture able to generate both 2D dungeons platformer levels and NPC behaviors. Our first results show that our game AI planning architecture represents a robust and online co-creative tool to automatically synthesize a wide range of 2D stable and solvable levels that provide an enjoyable and entertaining game immersion. We believe this architecture can be applied to produce various others structures for other game genres.

REFERENCES

[1] E. Hauck and C. Aranha, "Automatic Generation of Super Mario Levels via Graph Grammars," *IEEE Conference on Games*, pp. 297-304, 2020.

[2] A. Liapis, "10 Years of the PCG workshop: Past and Future Trends," *International Conference on the Foundations of Digital Games*, New York, USA, no. 96, pp. 1–10, 2020.

[3] A. Khalifa, M. C. Green, G. Barros, and J. Togelius, "Intentional computational level design," *Genetic and Evolutionary Computation Conference*, pp. 796-803, 2019.

[4] Y. Du et al., "Automatic level Generation for Tower Defense Games," *IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference*, pp. 670-676, 2019.

[5] I. Dart and M. Nelson, "Smart terrain causality chains for adventure-game puzzle generation", *IEEE Conference on Computational Intelligence and Games*, pp. 328-324, 2012.