Portfolio Element Two: Procedural Building Generation for the Game "Neighbours from Hell: Season 1"

Temur Kholmatov, Innopolis University

October 2020

1 Introduction

1.1 Background

The Neighbours from Hell is a puzzle strategy game that was released in 2003 (Wikipedia contributors, 2020). In this game, the gamer plays for Woody who participates in a reality show. The main goal of the show is to perform bad tricks upon Woody's neighbor and, this way, complicate his life. The game consists of several seasons. In this project, we focus on the first season where the game happens in a building with several floors and rooms where you should complete some tasks. The project attempts to create a generator of building with rooms and, especially, the doors distribution with the location logic specific exactly to this game.

The source code of the project with description is available here: `https://github.com/temur-kh/procedural-building-generation`.

1.2 Taxonomy

According to the taxonomy (Togelius et al., 2011), one may define the following distinctions of the algorithm:

- Offline. The algorithm itself is not so complex and can be modified to run online. However, the game itself does not require an online generation of rooms. That is why one may run this algorithm once before a level of the game.
- Necessary. In such games as "Neighbours from Hell: Season 1", a building with rooms is one of the most important assets.
- Small Control. There are comparable a few parameters that can be defined by a user.
- Generic. The player does not affect the process of building generation.
- Stochastic. The method implies the usage of random values.

- Constructive. The method creates a building in one iteration without any search-based approaches.
- Automatic. A human designer is not involved in the process of building generation. However, human intervention could help in the decoration of the building with some home decoration assets and corresponding game tasks. Also, a human is involved in the evaluation task if no automatic tests are used (see Section 3.1).

2 Method

The method is based on the usage of a modified depth-first search through a randomly generated graph of floors and rooms. The general pipeline of the proposed method is defined to be like this:

1. Generate floors and assign them random sizes concerning user-defined constraints.
2. Generate rooms on each floor and assign them random sizes concerning user-defined constraints and constraints of their floors.
3. Use a depth-first search to construct the paths through all rooms and create cycles in the graph with some probability.
4. Use prefabs of doors and a wall from a prototyping pack to generate a building according to the constructed graph.
5. Evaluate the results either manually or by automatic tests.

2.1 Building graph

The process of the graph generation for a building is described with the following steps:

1. Create several floors. Each floor has a starting position and size. A starting position is shifted from the position of the lower floor for a random number of units. The maximum step of the shift is a user-defined parameter. The size of each floor is a random value in a range of user-defined constraints.
2. For each floor, create rooms. The number of rooms is a random value in a range of pre-computed constraints that depend on the floor size and user-defined parameters of the maximum and minimum room size. Each room has a starting position concerning its floor's initial position and the end position of the previous neighbor room. The size of each room is a random value in the defined constraints. Moreover, the sum of sizes of rooms on the same floor is equal to the size of the floor.

2.2 Doors distribution

There are two types of doors: doors between neighbor rooms on the same floor and doors connecting two rooms vertically. The second type of doors requires special requirements to be satisfied:

- Two doors should be generated: one for each connected room.
- Doors should connect room that are directly above one another. Thus, no additional floors should be located between those rooms.
- Doors should not be placed on the last unit positions of a room.
- Each door connects its room with only one room.

The doors are randomly generated on a way of the depth-first search algorithm. Additionally, the algorithm may create connections between already visited rooms, thus, leading to cycles in the graph. This operation is done with some probability which is a user-defined parameter.

2.3 Building construction

For the building construction, we need three prefab objects:

- A door of the first type (connections between neighbor rooms).
- A door of the second type (vertical connections).
- A wall for the rest of the building.

We use ready-made prefabs from a prototyping pack. The process of objects instantiating is quite straightforward, and only two geometric operations are involved: translation and rotation.

3 Evaluation and Results

3.1 Evaluation Metrics

In order to satisfy the user's evaluation process, a generated building should pass so called definitions of done:

- The number of generated floors is equal to the corresponding user-defined parameter.
- The sizes of floors are in the corresponding user-defined constraints.
- The sizes of rooms are in the corresponding user-defined constraints.
- The floor shifts are in the corresponding user-defined constraints.
- Each door of the second type meets its requirements (refer to Section 2.2).
- The user can reach every room on each floor starting from the initial position. Thus, the graph of room is connected.

As long as the size of a building is feasible, it is possible to check the generated building by these criteria manually. One will not hold this process before each game, so manual work would be required if the levels were generated before the game production. However, if you need to generate large buildings or the online building generation is required, one may write assertion tests for the application.

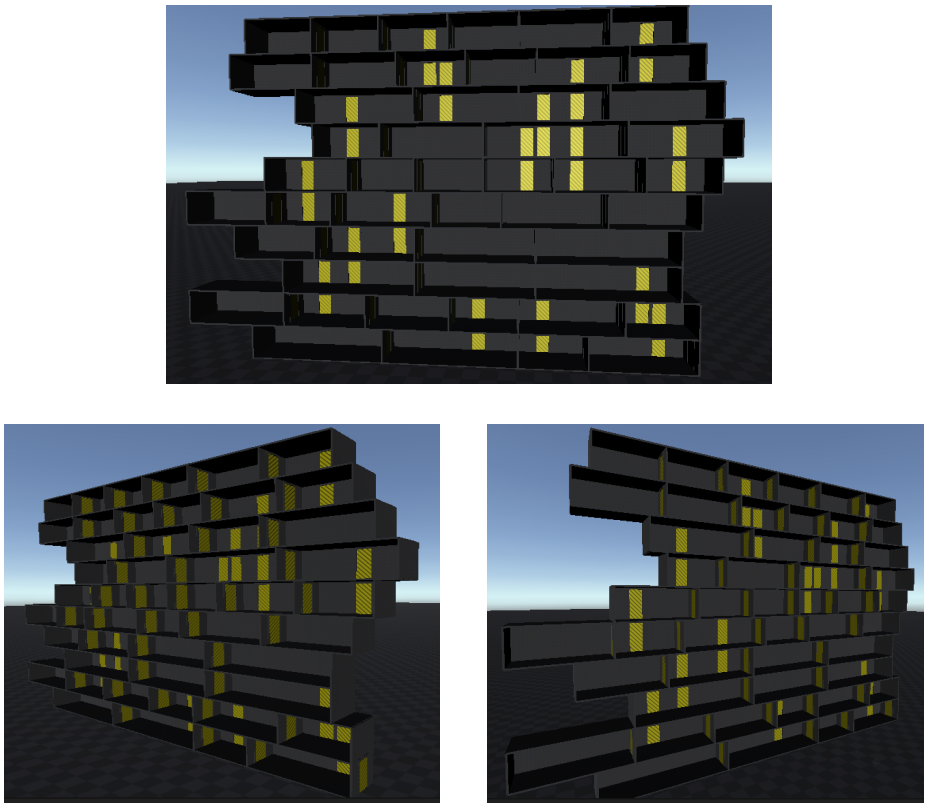

Figure 1: An example of a generated building from different views.

3.2 Results

In Figure 1, you may observe an example of a generated building. Here is the list of parameters used for its generation:

- The random seed = 42.
- The number of floors = 10.
- The minimum and maximum floor sizes are 10 and 20 units, respectively.
- The minimum and maximum room sizes are 4 and 8 units, respectively.
- The maximum floor shift allowed is equal 1.
- The probability of creating of a cycle in a graph of rooms is equal 0.25.

4 Further Work

The project is useful as a basis for the real game "Neighbours from Hell" with level generation. However, one will need to create a generator of house decoration assets and, what is more difficult, implement the logic for interesting tasks and tricks. Another important thing for the game production would be to create an attractive design of a building because in this project we used simple prototyping game objects.

References

[1] Togelius, J., Yannakakis, G. N., Stanley, K. O., Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games, 3(3), 172-186.

[2] Wikipedia contributors. (2020, October 11). Neighbours from Hell. In Wikipedia, The Free Encyclopedia. Retrieved 19:23, October 13, 2020, from https://en.wikipedia.org/w/index.php?title=Neighbours_from_Helloldid=982953225