Modern Games AI Algorithms A1: Procedural Content Generation

Giulia Rivetti (s4026543)

09/03/2024

1 Introduction

This assignment deals with Procedural Content Generation (PCG), which indicates the automated creation of content, such as game levels, landscapes, characters, or items, using algorithms. We will see how to apply this approach in order to build a house in Minecraft, given a specific build area. The program scans this build area and tries to find the best location in which to place the house, without modifying the environment too much. Moreover, randomness was also introduced into the construction of the house, by changing its dimensions, colour and the furniture.

2 Inspiration

My design was inspired by the famous *Smurfs* houses, which are mushroom-like houses as can be seen in Figure 1. The external structure of my house is indeed a mushroom and then, for the style I decided to go for a "cottagecore" design, taking inspiration from some Minecraft houses, such as the one showed in Figure 2. In my house, the cottage-core aspect can be seen in the usage of wooden blocks and the presence of a patio.

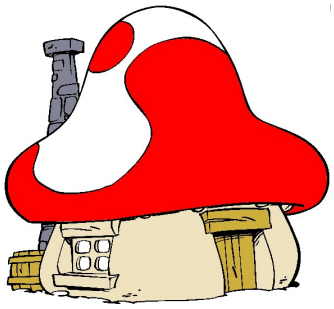

Figure 1: The Smurfs' house, which give me inspiration for the shape of my building.

3 Adaptability and Believability

3.1 Finding the best building region

Before actually building the house, an important part of the assignment is to scan the given Minecraft build area and find the best location in which to build: by this we mean that the house should adapt to the existing environment, by searching for a proper building region and by modifying as few blocks as possible. I will now explain how my implementation handles this part.
After retrieving the build area, the algorithm searches for all the possible regions within the build area. Each region has the same size, which I set to 22 based on the dimensions of the

Figure 2: Cottage-core house built in Minecraft that I took inspiration from, for the architectural style of my assignment..

house, and they are all square-shaped. At first, I've tried to implement the algorithm so that it was searching for regions of different sizes, but then I have noticed that the running time was increasing exponentially with the number of sizes of a region, and therefore it became clear that I needed to find a trade-off between getting regions of different size (thus increasing the chance of finding the best region) and decreasing the running time. I've decided to go for this second option since the algorithm is still able to find good building regions.
Once the algorithm has identified a region, it proceeds with the steps reported in Algorithm 1. By looking at the pseudocode, it can be noticed that the algorithm first computes the average

Algorithm 1 Pseudocode for the selection of a region

```
for each region do
    compute the average height of the region with the heightmap
    if average height is 63 then                          ▷ If we are the sea level
        if there is water in the region then
            discard the region
        else
            continue
        end if
    end if
    if average height is within the build area then
        compute the score for the region
        if score > 0 then
            store the region and its score into a list
        end if
    end if
end for
```

height of the region. Since this value indicates the height at which most blocks are within the region, it will be used as the starting height from which to build the house (if that region is selected in the end). The algorithm then proceeds to check if the average height is on the sea level; notice that 63 is generally the height at which water blocks are placed in Minecraft. If it is, then the algorithm checks if the region is actually containing water and in case it does, the region is discarded, because a house floating on the ocean wouldn't be particularly believable. After checking that the average height is within the bounds given by the y-axis of the build area, a score is computed for the region. The score is based on the number of blocks that need to be modified (added or removed) in order to clear the region from obstructing blocks and place the house there. This means that the best region that can be found, has the lowest score, since that would result in less modifications to the surrounding environment. Therefore, in the end the algorithm selects the region corresponding to the lowest score. However, before choosing this region as the place where to build the house, the algorithms proceeds with another check:

making sure that no other building is already in that region. Indeed by running the code multiple times, different houses are generated and therefore a certain region could already be occupied by another house. If this check is also passed, then the algorithm has found a place in which to build the house. Otherwise, the region with the second best score is selected to undergo this last check. This is done until a region that does not overlap with an already existing building is found.
There can however also be the case where the algorithm cannot find any suitable region to build the house: this can happen in a few cases, such as when the start of the build area is set too high and all of the existing blocks in Minecraft are below that threshold[1]; when there are already too many buildings in the build area and the algorithm cannot find enough space to build another one; or if the build area is mainly occupied by water. In this last case, if an area of size 23 cannot be found because all of the possible regions are actually floating on the water, then the algorithm just tells the user that he hasn't found a build area because of that problem.
I have decided not to put a threshold for the number of blocks that can be modified in order to build the house: indeed, even if the build area is in a particularly crowded environment, like a forest, the algorithm still flattens the terrain and builds the house. I've made this decision in order to show the adaptability of the algorithm to the given environment; however, this means that in some cases the algorithm may take a few seconds longer to clear the region and build the house.

3.2 Clearing region from obstacles

As already explained before, the score assigned to a region is based on the number of blocks that need to be modified in order to make the area flat and position the house there. After a region has been chosen as the actual building region where the house will be placed, the terrain is actually modified in order to both eliminate obstructing blocks and to add blocks so that the area will be flattened. For the purpose of making everything more realistic and believable, the algorithm checks for the kinds of blocks that are placed in the region, and it used those block types if new blocks are needed to be added. This means that if the region is in the desert where the blocks are made of sand, the blocks that will be added to make the area flat will be sand blocks.

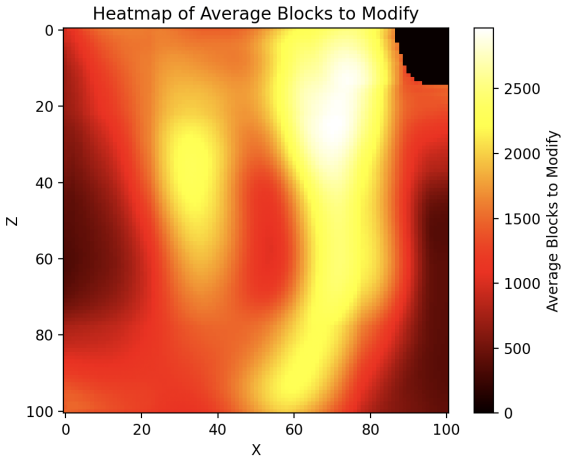

Figure 3: Heatmap of the terrain evaluation. The plot shows the number of blocks that need to be modified in order to build the house there. The x and y coordinates correspond to the coordinate of a certain block within the build area.

Figure 3 shows the heatmap representing the terrain evaluation done by the algorithm. Since the evaluation was done by assigning a score to a certain region and some regions can overlap with others, in order to assign a score to a specific block for the plot, I have taken the score of each region in which the block was contained, I've summed them up and then averaged the result. Notice that the region at position (x=100, z=0) is completely black because it indicates the presence of water (since a region with water is not taken into account it has a score of 0). Besides that, the best region is indicated by darker spots, like the one on the left around position (x=5, z = 60) or the one on the right around position (x=95, z=80). Indeed when I have obtained this specific plot, the house was then placed on the darker region on the left. It is important to remember than since in this case we are taking the average of the score of a single block, while the algorithm computes the score of an entire region in order

[1]When running the program remember to check that the lower y limit of the build area is not too high, otherwise no region will be found because it wouldn't make sense to place a house in the air

to place the house, there can be a slight difference between the best region indicated by the heatmap and the one obtained by running the code. In any case, the heatmap still shows that the house was placed in a position with a low number of blocks that need modification.

4 Variability and Randomness

Since the keyword of this assignment is Procedural Content Generation, the program shouldn't generate the same house over and over again at each run, but it should vary a little. In this section I explain which parameters and hence parts of the house randomly change at each run.

4.1 House dimensions and structure

Even if the build regions has a fixed size of 22 and therefore also the perimeter of the house, which is delimited by a fence, the actual house has variable dimensions. Since it is a mushroom house, it is composed of a cylinder that represents the stem, and an ellipsoid used for the cap of the mushroom (hence the roof). To establish the dimension of the house, I have considered the diameter of the cylinder: this parameter can take a value from 6 to 10 extremes included. By varying the diameter of the cylinder, also its height varies and since the size of the roof depends on the cylinder dimensions too, this means that also the size of the mushroom cap varies. Both smaller houses and bigger ones have a ground floor and an upper one; however, what mainly changes in their structure it's the presence of a second small mushroom that serves as an additional room on the ground floor. Indeed houses with a bigger diameter (either 9 or 10) are characterized by these additional room placed on the side of the main house, which is used as a sort of small greenhouse.

Figure 4: Example of a house of diameter 9 with an additional room and an outside swimming pool

Moreover, both external decorations and internal furniture depends on the diameter: if the diameter is odd, some blocks are placed twice in order to have symmetry, since in this case the side of a mushroom would have a length of 4. If instead the diameter is an odd number, blocks are only placed ones (such as the stairs leading to the patio), the doors, the chest, ...).

4.2 House colours

The house also varies in colour. In particular, there are two different colours possibilities: either a house with a white stem and dark oak wood decorations (such as the patio or the balcony), or a mushroom with a brown stem and with lighter oak wood decorations to contrast with the dark colour of the stem. I've decided to generate only two possible colour combinations because, since the house is a mushroom, it only made sense to use mushroom blocks to build it, hence the "mushroom stem" and the "brown mushroom block" for the stem. The result of varying house dimenion, structure and colour can be seen by looking at Figures 4, 5, 6.

4.3 House interiors

Another aspect that randomly varies is the furniture of the house. The ground floor is different for smaller and bigger houses as showed in Figures 8, 8, 9, 10: the placement of table and chair can randomly vary, as well as the presence of a window or a windowsill decorated with candles.

Figure 5: Example of a house of diameter 8 with an outside swimming pool

Figure 6: Example of a house of diameter 6 with an outside hortus.

Figure 7: Ground floor of house with a diameter of 6. It has a single table and chair, a lantern and two windowsills with candles.

Figure 8: Ground floor of house with a diameter of 7. It has a single table and chair, a lantern and one windowsill with candles.

Moreover in bigger houses the table is bigger and the ground floor is also characterized by a fireplace, which is not present in houses with a diameter lower than 9.
As for the upper floor, which is showed in Figures 11, 12, 13, 14,where the bedroom is situated, randomness has been introduced by changing the disposition of the furniture (either on the left wall or on the south wall) and the colour of the bed, and variability can be seen in the placement of different objects and blocks, such as windows, a desk with a library or flowers.

4.4 Garden

The house is surrounded by a small garden, which I decided to decorate with some flowers, whose colour changes randomly between the different available tulips colours present in Minecraft (see Figure 15, 16). What also varies randomly is the presence of either a swimming pool or a small

Figure 9: Ground floor of house with a diameter of 9. It has a single table and chair, a lantern and a fireplace.

Figure 10: Ground floor of house with a diameter of 10. It has a 4 tables and chairs, a lantern and a fireplace.

Figure 11: First floor of house with a diameter of 6. It has a double bed, two sets of windows, a chest, a bookshelf and flowers as decorations.

Figure 12: First floor of house with a diameter of 8. It has a double bed, two sets of windows, a chest, a bookshelf and flowers as decorations.

Figure 13: First floor of house with a diameter of 9. It has two sets of windows, a double bed, a bedside table with candles, a chest, a library, a desk and a chair.

Figure 14: First floor of house with a diameter of 10. It has two sets pf windows, a double bed, two bedside tables with candles, a library, a desk, a chair and a double chest.

Figure 15: Figure showing more in detail the garden with the hortus

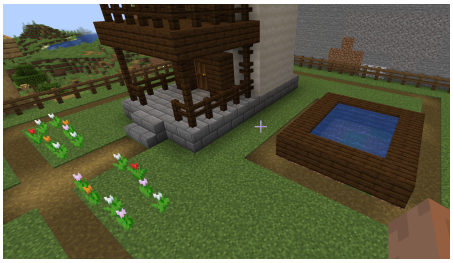

Figure 16: Figure showing more in detail the garden with the pool.

hortus; these two items also change in position: for smaller houses they are placed in the center (considering the x-axis) of the perimeter, right next to the house. However, since bigger houses have an additional room on the side, in this case the pool/hortus will be placed more to the front.

5 Conclusion

In this assignment I have showed the result of employing Procedural Content Generation in order to build a house in Minecraft. The resulting algorithm scans the build area and finds a good region where to place the house in favour of adapting to the existing environment in a believable way. Moreover, with the introduction of randomness and variability, the algorithm is also able to generate different houses with a similar style at each run.