Posudek diplomové práce

Matematicko-fyzikální fakulta Univerzity Karlovy

Autor práce	Jakub Mifek		
Název práce	Procedural Generation of Minecraft Villages utilizing the Wave Function Collapse Algorithm		
Rok odevzdání	2022		
Studijní program	Informatika	**Studijní obor**	Umělá inteligence

Autor posudku	Adam Dingle	**Role**	Oponent
Pracoviště	KSVI		

Text posudku:

This master's thesis explores the use of the relatively new Wave Function Collapse (WFC) algorithm in procedural content generation for the popular game Minecraft. I was not aware of the WFC algorithm before reviewing this work, however I agree that it is an interesting algorithm that has some promise in this area.

In section 1 "Background" Jakub gives an overview of the WFC algorithm and of Minecraft. I appreciated the overview of the algorithm in section 1.2 including pseudocode. However I found that the description here was not quite precise enough for me to understand the algorithm completely, and so I had to read the article "WaveFunctionCollapse is Constraint Solving in the Wild" by Karth and Smith to solidify my understanding. It would have been helpful to illustrate the algorithm's behavior in detail on a small specific example (like in the Karth and Smith paper). The pseudocode includes the phrase "minimal entropy of patterns", but this concept is not explained at all in the text. In 1.2.2 "Models" I'm not sure what is meant by "M << N"; perhaps this is a typo and should simply be "M < N"?

I installed Minecraft on my machine and was able to install and run Jakub's Minecraft mod. I ran all the commands in the user guide and confirmed that they worked. I had some trouble at first, since I found that the mod was unable to replicate the "house" template. I emailed Jakub and he explained that I should be using the "test4" template instead, which is a bare-minimum template (as described in 6.1.3 "Diversity"). That worked better. However, the name "test4" is cryptic and it was not obvious that I should be using this template; I wish it had a name such as "simple_house" or "basic_house".

I generated several villages using the "gv" command. The largest village I made had dimensions 100 x 10 x 100 (the units here are Minecraft blocks) and had 20 houses. It looked basically reasonable. However, the structure of the houses was quite simple: most were simple rectangles or T shapes. It would be easy to generate houses with these shapes even without WFC, of course. The houses in section 6.2.1 "House generation showcase" are more diverse than the ones that I saw when I used the "gv" command; I'm not sure why.

I found the house generation to be quite slow – it sometimes took minutes on my machine to generate even a single house. That seems to be a weakness of the WFC algorithm in 3 dimensions. In section 3.3.2 "Performance" Jakub explains that his implementation is an

order of magnitude slower than Maxim Gumin's original WFC implementation. That seems disappointing, but I'm glad that Jakub attempted to profile his program and explain the performance difference. Apparently it is mostly due to the general nature of his implementation.

In section 6.1.3 "Diversity" Jakub explains that WFC on its own can't replicate templates that have a lot of detail, and suggests several solutions to the problem. I think approach 3 (splitting the generation into multiple phases) seems promising, and I think the 2-D floor plans it generated look interesting. The text says "Because the floor plan 2D patterns allowed for many neigbours, the algorithm often reached a contradiction when the result was applied to 3D generation". It's not completely clear how the 2D result was "applied" to 3D generation here; I wish the text explained this in more detail. The text also says that each pixel in the floor plan could represent the IDs of all the vertical blocks above it, but that "that approach gets us back to similar complexity of the original generation". It's not clear to me what is meant by "complexity" here (run time? failure to converge on a solution?) A specific example or more descriptive text would have been helpful here.

I looked a bit at the Kotlin code in Jakub's implementation. It seems good that the implementation is modular: at a high level, it consists of several libraries (WFC, village library, Minecraft mod) and also the WFC library has several layers (core algorithm, models, adapters). On the other hand, it seems that this modularity had a performance cost. In some places I found the code hard to read since many lines were long (well over 100 characters, so they didn't fit in an editor window) and some methods were very long as well (e.g. MinecraftWfcAdapter.imitate).

Overall this thesis seems like a successful exploration of applying WFC to procedural content generation in 3 dimensions. The algorithm certainly has its limitations: it's pretty slow and seems to be able to replicate only simple structures such as the house template "test4" or the "silo" or "moderna2" templates. That was a challenge for this thesis work. However Jakub is aware of these limitations and explains them, and has tried to find ways to work around them, e.g. via his preprocessing and postprocessing phases in house generation, or by splitting the generation process into multiple phases.

In conclusion, I recommend this work for a thesis defense.

Práci doporučuji **k obhajobě.**

Práci nenavrhuji **na zvláštní ocenění.**

Pokud práci navrhujete na zvláštní ocenění (cena děkana apod.), prosím uveďte zde stručné zdůvodnění (vzniklé publikace, významnost tématu, inovativnost práce apod.).

Datum 19.1.2022 **Podpis**