VIDEO PLAN

**Introduce My Project**

Hello, my name is Edgar and this is how I created my own algorithm which procedurally generates levels that one would commonly find in roguelike video games.

For a bit of context, procedural generation is a technique used to create content algorithmically with the help of randomization.

Before, I explain how my algorithm works, let me show you how the design came to be.

**Refer to how I looked into WFC and CA**

When I began this **process**, I had no idea on how procedural generation worked. So therefore, I began researching different ways in which proc gen is used within the industry.

Through my research I found a variety of techniques, however there were two that seemed the most interesting to me – them being wave function collapse and cellular automata.

**Explain what I took from each**

See WFC works through the use of a database and a starting point. Looking at the starting point, **one** selects a path and looks at which pieces in the database fits.

**From** here we select one at random and Then the process is repeated until all of the pieces have been **generated**.

As for cellular automata, the algorithm works through a grid of cells which have different states. **For** example, starting with a cell that can either be a wall or a floor. The algorithm then generates a **grid** with a bunch of cells which are randomly assigned their state.

Then one must go through each cell and check their **neighbours** --- if most of them are walls then the cell will also turn into a wall, otherwise it will become a floor. Eventually, after a few iterations the cells will create groups of the same states and form some **formation**.

So by learning how both these algorithms work, **I was** able to take the grid structure form cellular automata and create a database like WFC does to generate some rooms. **By then** using the neighbour check from CA along with the piece selection from WFC, I was able to create a new algorithm.

Let me show you how it works:

**Showcase my final algorithm**

To start with we set a grid width and height, for example 10 by 10.

**The algorithm** then will generate an invisible grid and plop the beginning room in the middle cell.

**Next up** we choose a random path from the available ones and we c**hec**k if the cell in that path is clear.

**If so we** can now spawn a room that fits. Just like matching a puzzle piece.

**Now** we move unto the new piece and we choose a free path. From here we do the same – Check the new spot, and get a fitting room.

**Now** we try to do the same but the spot isn’t free. Therefore, instead of spawning a room, **we spawn a door.**

Since there are no more paths in this branch, **we** go back to the spawn room and repeat the process.

Ultimately the level will finally generate itself and we are left with something like **this**.

The algorithm itself is a bit more complex, with more neighbour checks to avoid rooms leading into walls, the void, and other issues. But at its most basic level this is how I was able to combine my learnings from the two techniques and create my **own**.