Content Generation For Game Design

Understanding Generative Space

Mike Cook
@mtrc

Download the Processing sketch: **github.com/possibilityspace**

Today’s Aims

- Understanding spaces - what is a *possibility space* and a *generative space*?
- How does our choice of generative algorithm affect how we explore possibility spaces?
- What kinds of tradeoff do we make when designing procedural generators?

Download the Processing sketch: **github.com/possibilityspace**

Possibility Space and Generative Space

Download the Processing sketch: **github.com/possibilityspace**

The **generative space** of a procedural generator is a set containing every piece of content **the generator is capable of producing**.

Generative Space - Minecraft

- Each Minecraft world has its own 64-bit seed.
- So, the total number of *unique* Minecraft seeds is:

$$2^{64}$$

$$= 18{,}446{,}744{,}073{,}709{,}551{,}616$$

The **possibility space** for a type of game content is a set containing every piece of content **that it is possible to express in this form**.

Possibility Space - Minecraft

- A Minecraft (PC) world is 32,000,000 blocks wide, 32,000,000 blocks long, and 256 blocks tall.
- So, the volume of a single Minecraft world is:

$$= 262{,}144{,}000{,}000{,}000{,}000$$

- For each block, there are 16 (say) possibilities…

$$16^{262{,}144{,}000{,}000{,}000{,}000}$$

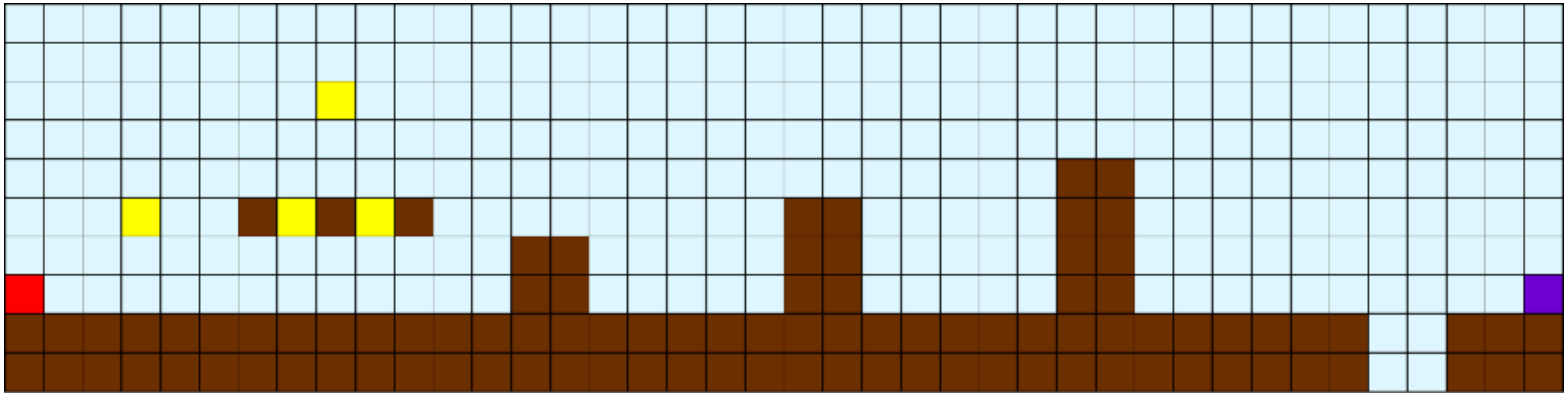
Summary

- Generative spaces contain all the things a generator can produce
- A possibility space contains all the possible pieces of content we can think of or represent
- Possibility spaces for game content are often ***much bigger*** than a generative space

Case Study
Platformer Level Generation

Download the Processing sketch:
github.com/possibilityspace

Download the Processing sketch: **github.com/possibilityspace**



As we look at each procedural generator, think about these questions:

- How big is the *generative space* of this generator?
- How often will this generator make something *bad*?
- How often will this generator make something *good*?
- How often will this generator make something *surprising?*

Download the Processing sketch: **github.com/possibilityspace**

Example 1: Randomness

Generation Method:

Every tile is randomly assigned a tile type, with equal chance of any tile occurring.

```
//For every tile in the map...
for(int i=0; i<levelWidth; i++){
 for(int j=0; j<levelHeight; j++){
   //Randomly place one of the tile types here
   level[i][j] = int(random(3));
 }
}
```

How big is this generator's *generative space* ?

Download the Processing sketch: **github.com/possibilityspace**

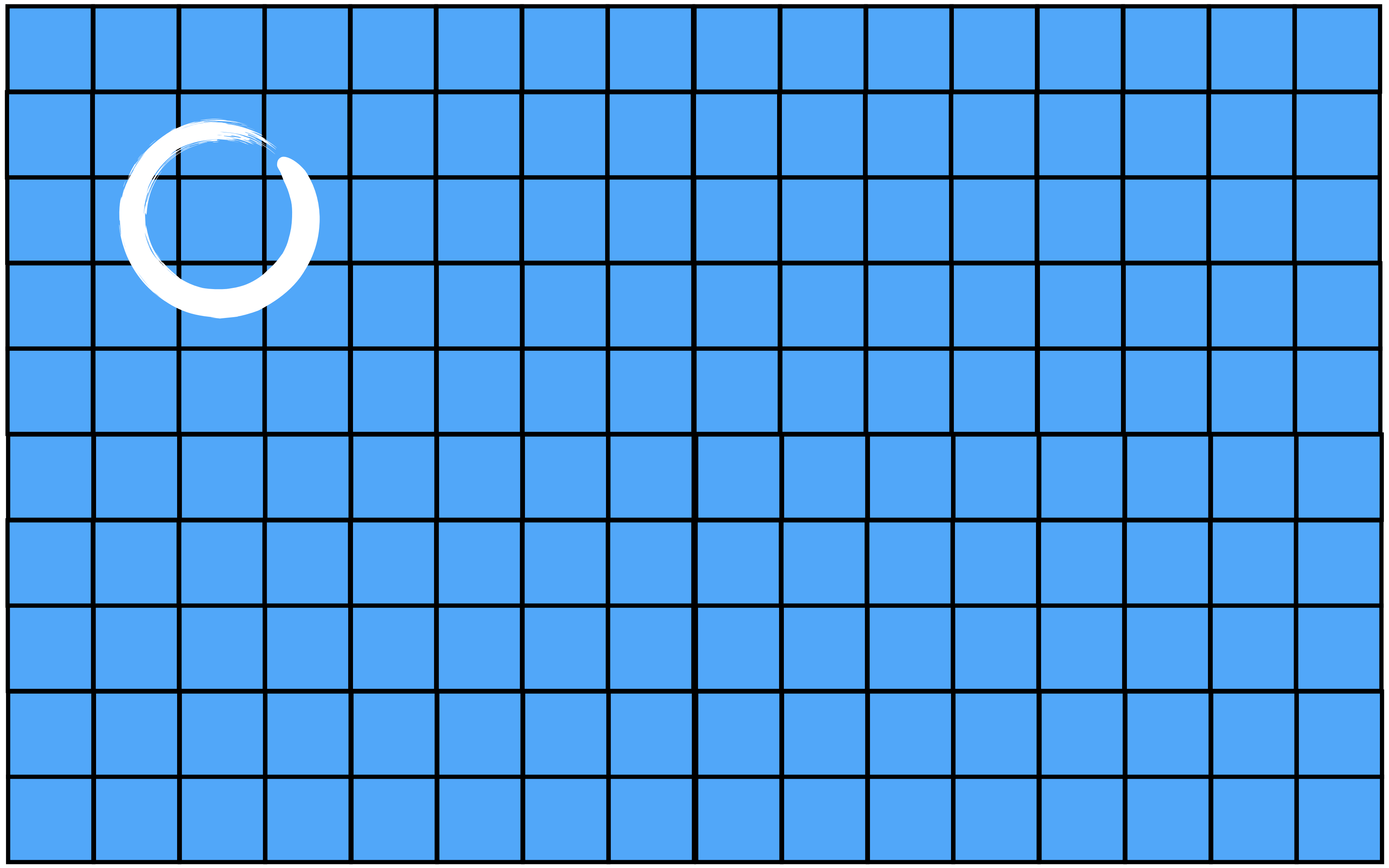
Example 2: Shapes

Generation Method:

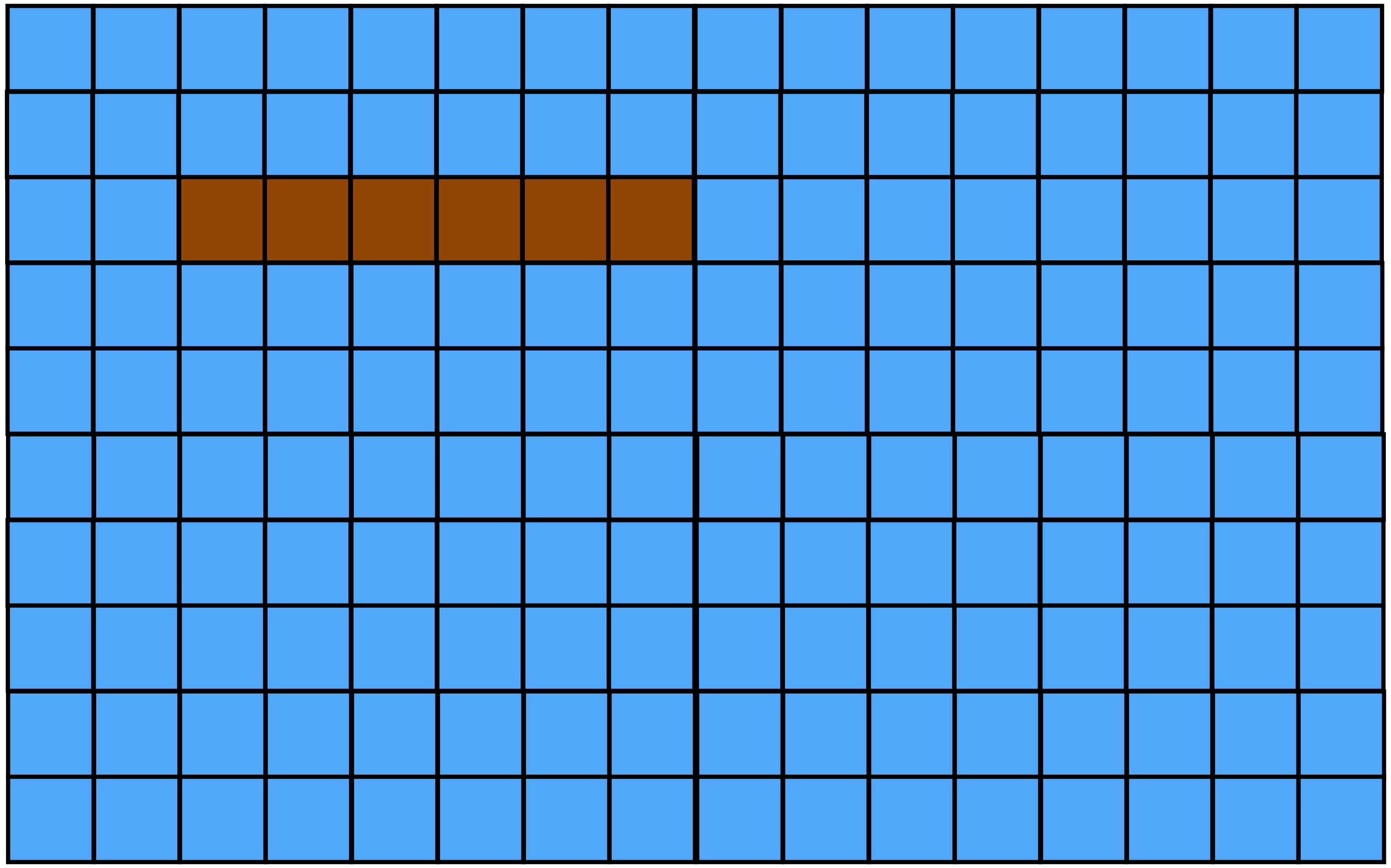
The generator draws a number of lines and blocks on the level, of random tile types. The number of shapes drawn is controlled by a variable.

```
//Place a number of shapes up to the limit
for(int p=0; p<numberOfShapes; p++){
  switch(int(random(3))){
    //Place a horizontal line
    case 0:
      //...
    //Place a vertical line
    case 1:
      //...
```

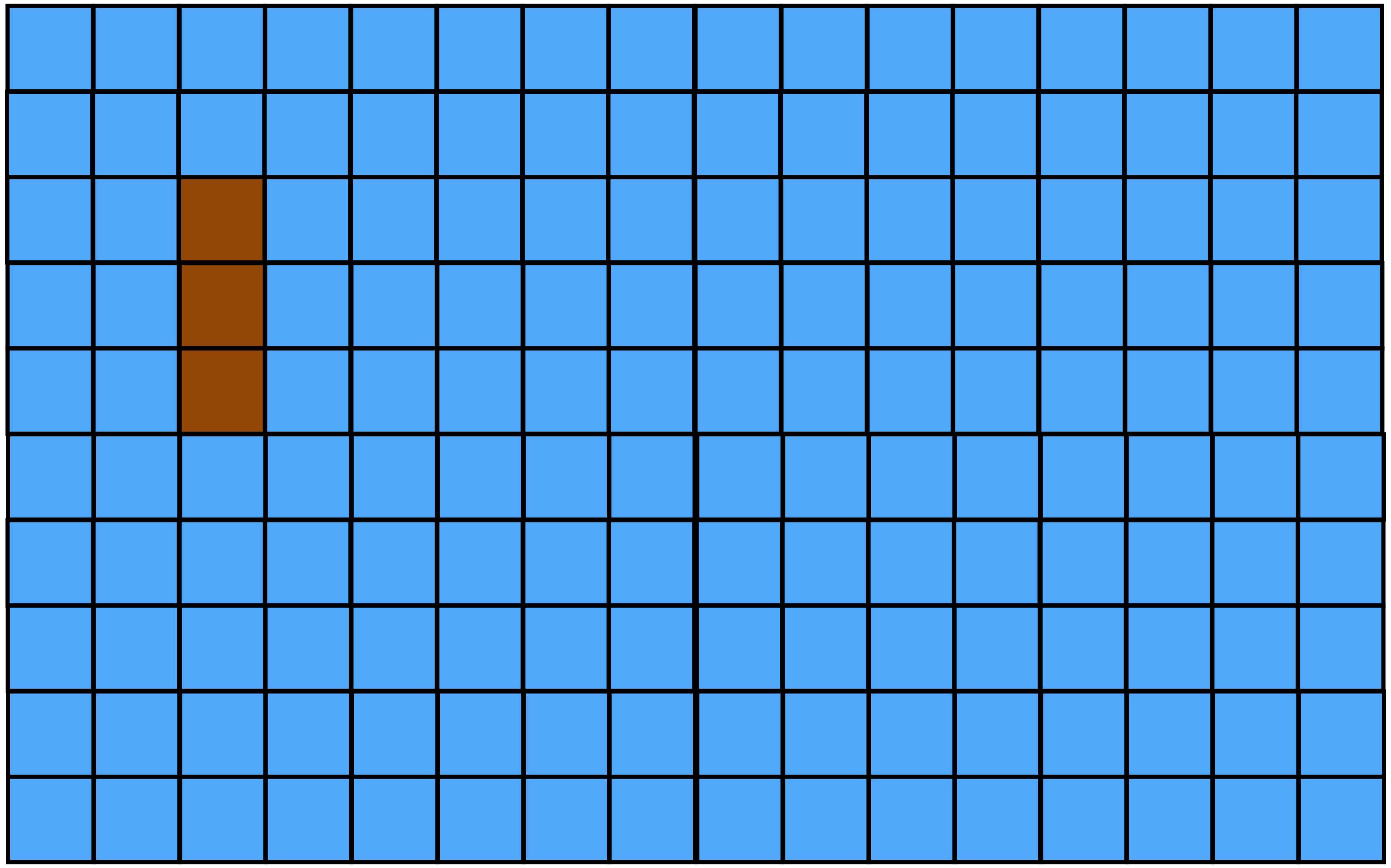
1. Pick a point on the level



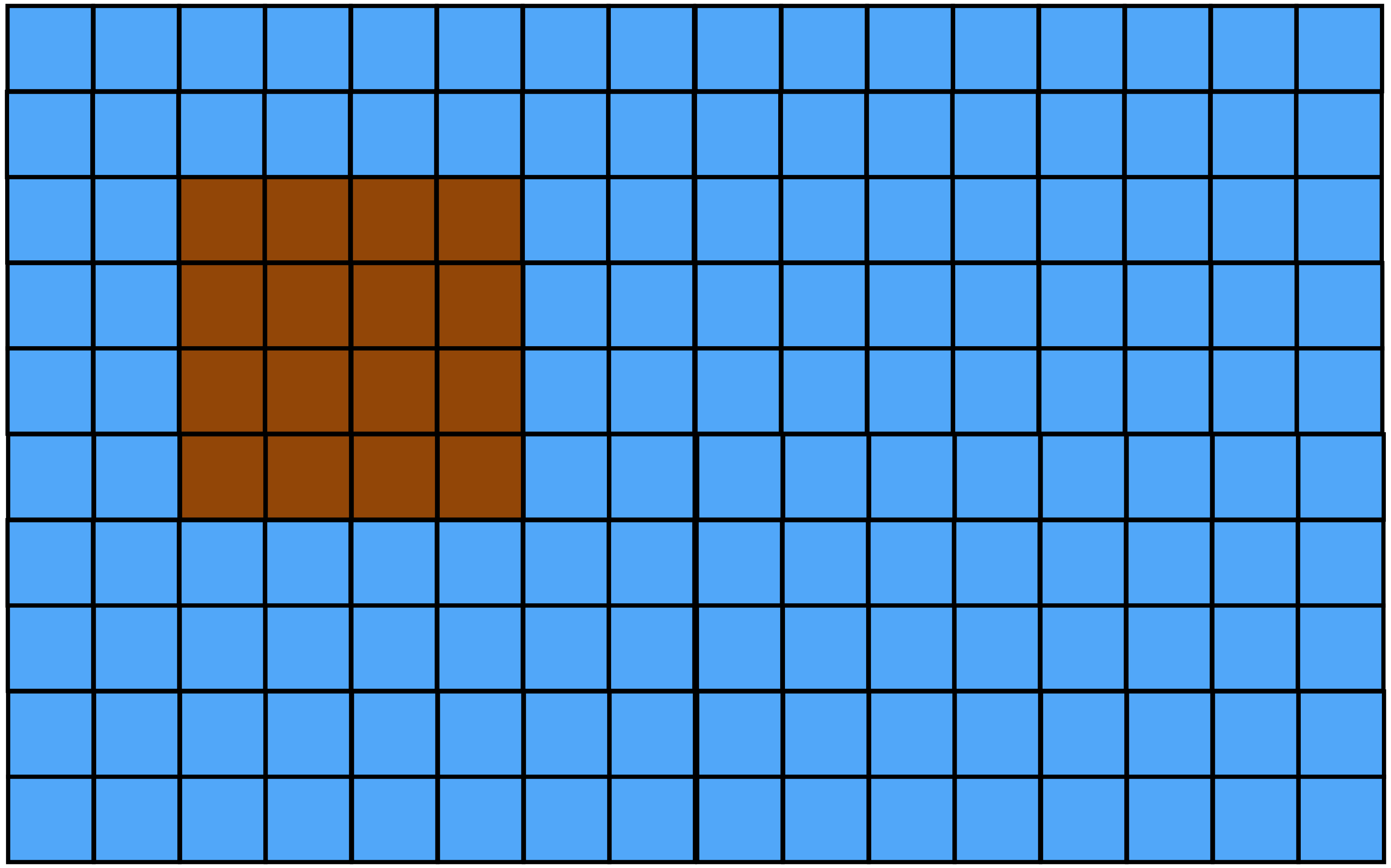
2. Pick a shape to draw



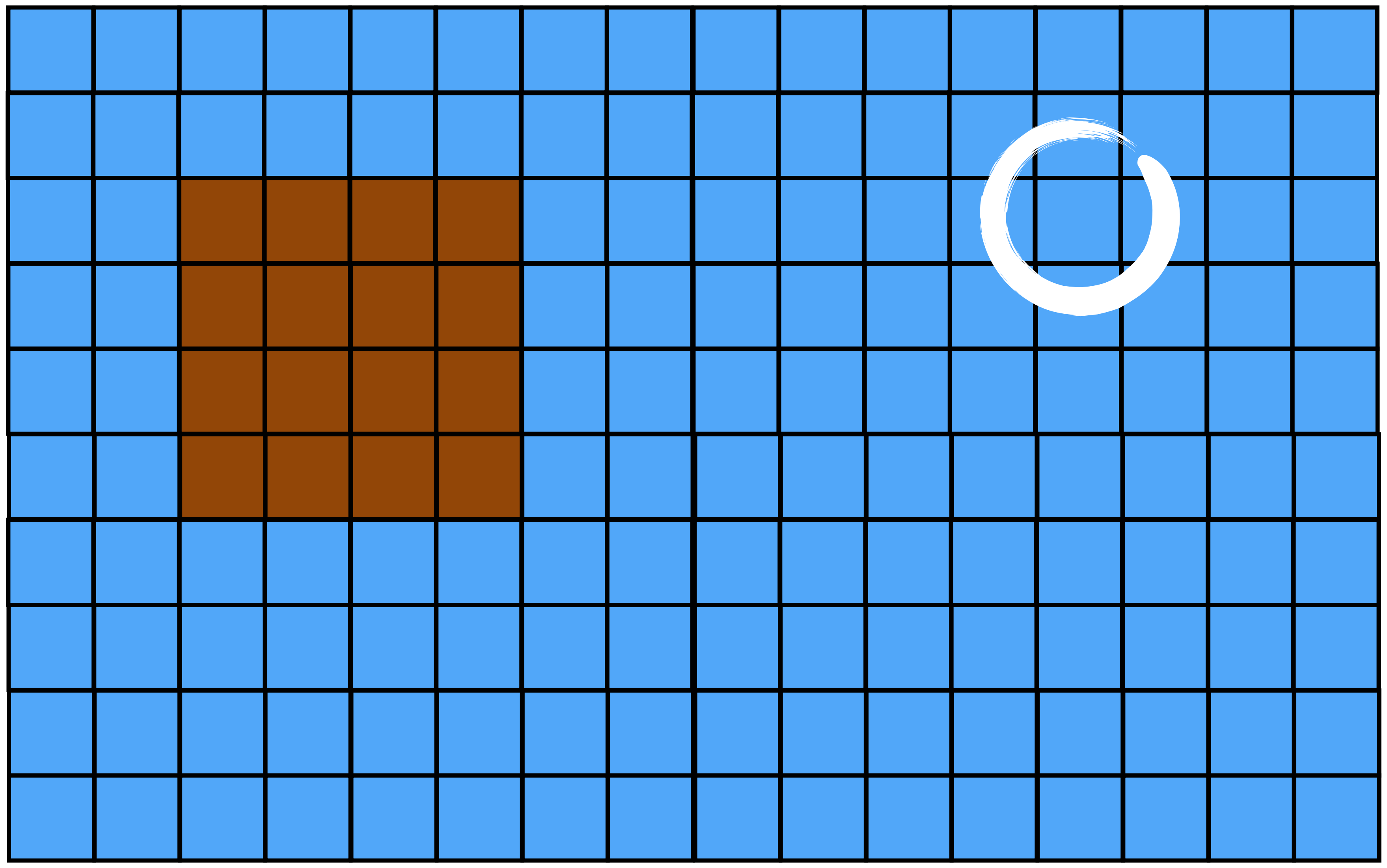
2. Pick a shape to draw



2. Pick a shape to draw



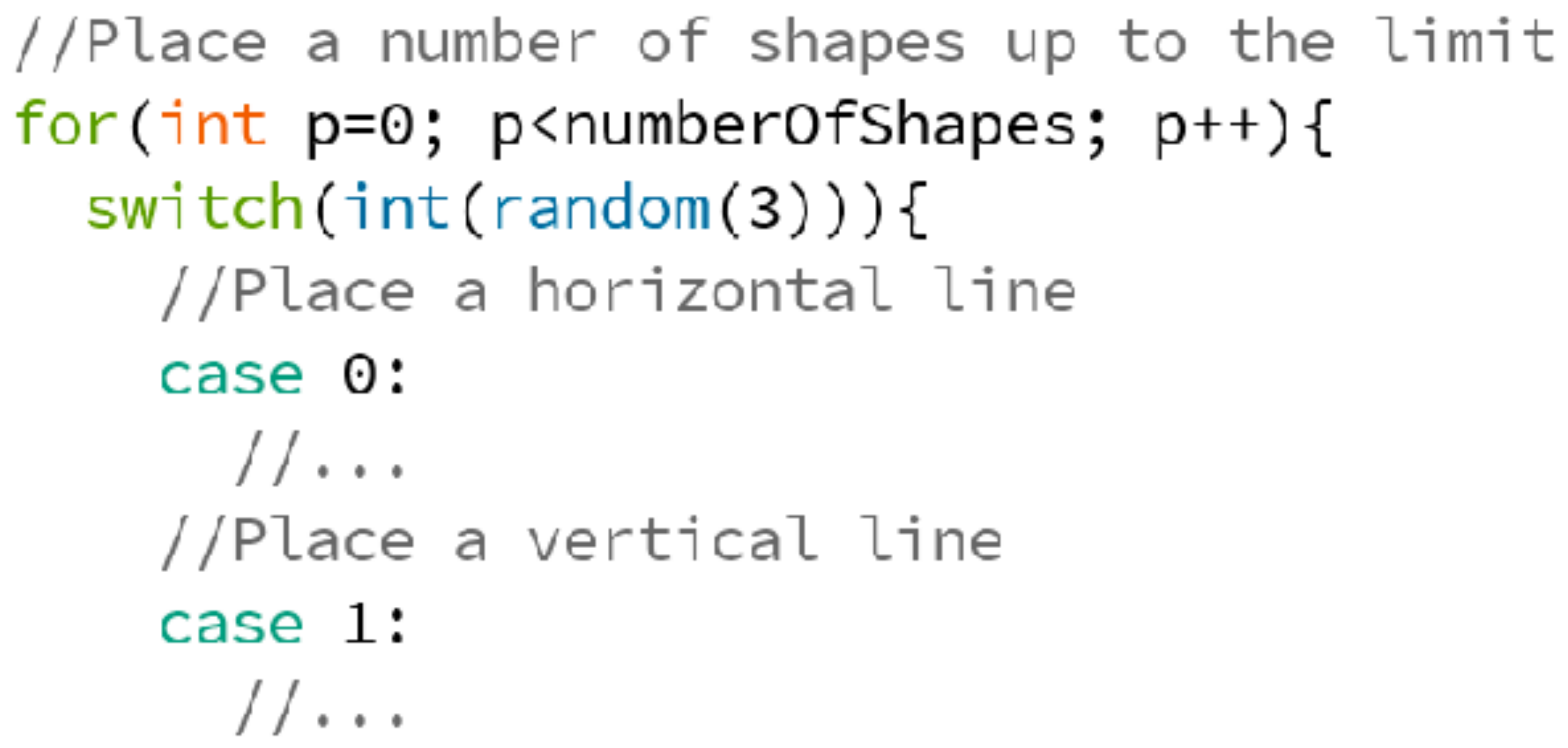
3. Repeat *n* times



Example 2: Shapes

Generation Method:

The generator draws a number of lines and blocks on the level, of random tile types. The number of shapes drawn is controlled by a variable.



```
//Place a number of shapes up to the limit
for(int p=0; p<numberOfShapes; p++){
  switch(int(random(3))){
    //Place a horizontal line
    case 0:
      //...
    //Place a vertical line
    case 1:
      //...
```

How long does it take to make a playable level?

Example 3: Shapes 2

Generation Method:

The same as Example 2, but this time with a smaller number of shapes drawn.

How long does it take to make a playable level now?

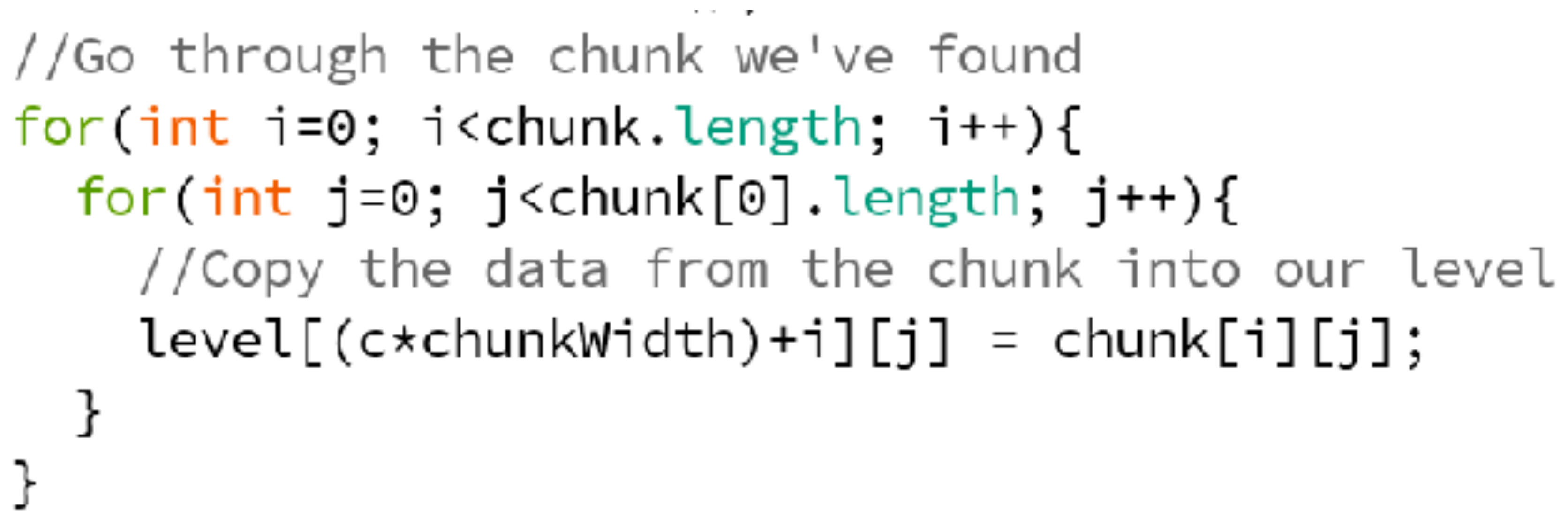
Which generator has a bigger generative space?

Which one is ‘better’? Why?

Example 4: Chunks

Generation Method:

Choose a random hand-designed level chunk, paste it into the level, repeat until the level is full.



```
//Go through the chunk we've found
for(int i=0; i<chunk.length; i++){
  for(int j=0; j<chunk[0].length; j++){
    //Copy the data from the chunk into our level
    level[(c*chunkWidth)+i][j] = chunk[i][j];
  }
}
```

How long does it take to make a playable level now?

How often would this generator surprise you?

Example 4: Humans

Generation Method:

Pay someone to make levels.

Which of our other generators contain these levels?

Summing Up

- *Generative spaces* are a way of discussing what things our generator can produce.
- *Possibility spaces* are a way of thinking about all the possible things we can represent in our game
- Bigger generative spaces contain lots of surprise, novelty, and variety - but can also contain more junk.
- Smaller generative spaces are more controlled and reliable, but can be bland and repetitive.
- Balancing these two ideas is the art of generation!