An infinite Minecraft house generator

Houses

Before you start:

- Launch Minecraft and create a new world.
- Launch Geany

Step 1 Make a roof

- In Geany, from the file menu select "new", and then save as `house.py` in the "pi" directory.
- We need to import the dBsCode commands we'll be using and clear an area in the Minecraft world for working in. Create this program:

```
from dbscode_minecraft import *
bulldoze()
```

- **Test** Press F5 to run the program (this will also save your program for you). After a few seconds you should see a "flatworld" type of environment.
- Lets start with the roof. Build a prism by adding this to the end of your script:

```
toblerone(WOOD,point(0,5,0),point(10,6,8))
```

- **Test** Press F5 and navigate to the centre of the world (using the coordinates at the top left of the Minecraft window). You should see a toblerone shape. This shape is drawn from block position 0,5,0 and is of size 10,6,8.
- We now need to hollow out the shape we've just drawn in order to make space for the walls:

```
toblerone(AIR,point(0,3,0),point(10,6,8))
```

This shape has the same dimensions but drawn 2 blocks lower, and as the material is set to AIR, it "cuts way" from the previous shape, leaving a roof structure.

Step 2 Build the walls

- Add this line so it comes after the roof toblerones:

```
box(BRICK_BLOCK,point(2,0,1),point(7,7,6))
```

- Press F5 to check this - it fills the area below the roof but we have a hole just underneath the top. We can fill this with another box:

```
box(BRICK_BLOCK,point(4,7,1),point(3,2,6))
```

- This house is not much good as it's solid, so we can hollow it out with an air box in the middle:

```
box(AIR,point(3,0,2),point(5,7,4))
```

- You can check this worked by destroying a few bricks and breaking in. Lets quickly add a door:

```
box(AIR,point(4,0,1),point(3,4,1))
```

Step 3 Make a house function

Functions are one of the founding ideas behind programming - they can amplify your actions, and allow you to solve difficult problems by breaking them into small ones.

- We use 'def' to build functions. Collect together all the code you've just written so it looks like this - the spaces at the start of the lines are important - there are 4 of them, or just press the "tab" key once:

```
def house():
    toblerone(WOOD,point(0,5,0),point(10,6,8))
    toblerone(AIR,point(0,3,0),point(10,6,8))
```

```
    box(BRICK_BLOCK,point(2,0,1),point(7,7,6))
    box(BRICK_BLOCK,point(4,7,1),point(3,2,6))
    box(AIR,point(3,0,2),point(5,7,4))
    box(AIR,point(4,0,1),point(3,4,1))
```

Now we can "call" this function by simply adding this to the bottom of your program (there shouldn't be any spaces before this):

```
    house()
```

Press F5 - your house should appear as normal. So far so good, but we haven't found out what the point of the function is yet...

- We can add a position "parameter" to the function. We use parameters to pass information into a function by adding them to the brackets at the top and refering to them inside. Change your existing function and add "pos" to all of the positions of the shapes:

```
def house(pos):
    toblerone(WOOD,pos+point(0,5,0),point(10,6,8))
    toblerone(AIR,pos+point(0,3,0),point(10,6,8))
    box(BRICK_BLOCK,pos+point(2,0,1),point(7,7,6))
    box(BRICK_BLOCK,pos+point(4,7,1),point(3,2,6))
    box(AIR,pos+point(3,0,2),point(5,7,4))
    box(AIR,pos+point(4,0,1),point(3,4,1))
```

We can now pass in the position when we call the function, to draw a house at any position:

```
    house(point(5,0,10))
```

Add another two:

```
    house(point(-5,0,5))
    house(point(8,0,0))
```

- Add as many houses as you like.
- What happens if you make them overlap?

Step 4 Make your houses look different

- Lets change the function to add a 'roof' parameter so our houses can have different roof materials. Also swap the `WOOD` in the first toblerone to be `roof` :

```
def house(roof,pos):
    toblerone(roof,pos+point(0,5,0),point(10,6,8))
    toblerone(AIR,pos+point(0,3,0),point(10,6,8))
    box(BRICK_BLOCK,pos+point(2,0,1),point(7,7,6))
    box(BRICK_BLOCK,pos+point(4,7,1),point(3,2,6))
    box(AIR,pos+point(3,0,2),point(5,7,4))
    box(AIR,pos+point(4,0,1),point(3,4,1))
```

Now we need to add materials to your houses:

```
    house(COBBLESTONE,point(5,0,10))
    house(LAPIS_LAZULI_BLOCK,point(10,0,10))
    house(MELON,point(13,0,-8))
```

- Lets go further and change the shape of the houses, by adding height. We need to move the roof up and change the size of the walls:

```
def house(roof,pos,height):
    toblerone(roof,pos+point(0,5+height,0),point(10,6,8))
    toblerone(AIR,pos+point(0,3+height,0),point(10,6,8))
    box(BRICK_BLOCK,pos+point(2,0,1),point(7,7+height,6))
    box(BRICK_BLOCK,pos+point(4,7+height,1),point(3,2,6))
    box(AIR,pos+point(3,0,2),point(5,7+height,4))
    box(AIR,pos+point(4,0,1),point(3,4,1))

house(COBBLESTONE,point(5,0,10),1)
house(LAPIS_LAZULI_BLOCK,point(10,0,10),5)
house(MELON,point(13,0,-8),20)
```

The melon roofed house is a tall one!

- What happens if you pass in height as a minus number?

Step 5 For loops: making *loads* of houses

If we want to make loads of houses, writing them all manually as `house(blah blah)' is a pain. One way we can make this easier is by using a "for" loop. Delete the houses at the bottom of your program and add this:

```
    for i in range(0,5):
        house(MELON,point(i*10,0,0),10)
```

This will repeat the `house` line 5 times (make sure you add 4 spaces before), each time with i being a number described by the `range` function - so 0 to 5 in this case. We set the X position of the house by multiplying this by 10 (so each house in the row appears spaced by 10 blocks)

- Try changing 5 to something bigger!

Step 6 Randomness: making all your houses different

- Randomness is a surprisingly important area and is used a lot in computer games as well as programming in general. Here we can use it to make every one of our houses different. Change the `house` function call in the loop to:

```
for i in range(0,10):
    house(MELON,point(i*10,0,0),random_range(0,20))
```

`rand_range` provides a random number between 0 and 20. Try running it multiple times - it should change each time.

- We can also change the block material for each house using the function `choose_one` which randomly picks between parameters which you pass in - as it's quite long lets break the line to make it easier to read:

```
for i in range(0,10):
```

```
    house(choose_one(BRICK_BLOCK,COBBLESTONE,BEDROCK,SANDSTONE),
          point(i*10,0,0),random_range(0,20))
```

- Try changing the position of the houses with `rand_range`

Challenges

- Can you change the width and height of the houses?
- Add windows - using GLASS brick type.
- These houses definitely need chimneys too.