Z level changes

- altering the floor level between rooms can be visually effective
- we will look at two ways, simple and more advanced method

Simple: chisel floor level changes

- currently `pen2map` converts a penguin tower file into a doom3 map file
  - however map floor is completely level
  - it would be good aesthetically to introduce minor floor level changes between rooms
- start up `emacs` and press F7 and then press F10
- in `pen2map.py` search forward for `assignFloorLevel`
- notice how it is called from `generateMap`

Simple: chisel floor level changes

- it would be pretty easy just to adjust the floor level between two values (for odd/even rooms)
  - surprisingly effective
- a unit of 1 in a penguin tower map represents 48 inches in the doom3 world
- can you think of a better algorithm in which to change floor levels?

Breadth first search algorithm for greatest height difference between rooms

- ideally we would like to have a large z differential between rooms
  - however we have a constraint of, say a single step between adjacent rooms
  - of course rooms might have multiple doors, connecting to different rooms (or the same room)

Breadth first search algorithm to adjust floors

- using a breadth first search algorithm we can satisfy the constraints of having a difference of a single step between neighbouring rooms

Breadth first floor level algorithm

$HOME/Sandpit/chisel/python/pen2map.py

```
#
#  calcFloorLevel - starting at the room with the spawn point
#

def calcFloorLevel ():
    global minFloor, maxFloor
    s = getSpawnRoom ()
    rooms[s].floorLevel = 0
    lowerFloors (s)
    for r in rooms.keys ():
        if rooms[r].floorLevel is None:
            rooms[r].floorLevel = 0
        minFloor = min (minFloor, rooms[r].floorLevel)
        maxFloor = max (minFloor, rooms[r].floorLevel)
```

Breadth first floor level algorithm

- `$HOME/Sandpit/chisel/python/pen2map.py`

```
#
#  lowerFloors - starting at room, s, lower all neighbouring floors
#

def lowerFloors (s):
    visited = [s]
    queue = getNeighbours (rooms[s])
    level = -(floorStep * noSteps)
    while queue != []:
        nextLevel = []
        # lower each neighbouring room floor
        for c in queue:
            r = rooms[c]
            if r.floorLevel == None:
                r.floorLevel = level
                nextLevel += getNeighbours (r)
        # now move onto their neighbours
        queue = nextLevel
        level -= (floorStep * noSteps)
```

Breadth first floor level algorithm

- take a sheet of paper, draw a graph of nodes and their interconnections
  - run the code by hand and change the node floor levels accordingly

- or write a tiny python test program to verify the algorithm