Generating brushDef3s

- maps in doom3 are generated using `brushdefs`
  - a `brushdef` can be used to describe a convex object
  - the most common is the `brushdef3` which define a series of infinite planes
- these planes are (internally by the doom3 engine) culled at intersection
  - resulting in the convex polyhedra object

Tiny example: cuboid

```
brushDef3
{
  // floor of fbrick
  (0 0 -1 0) ((0.0078125 0 0.5) (0 -0.0078125 -1)) "textures/hell/cbrick2b" 0 0
  // ceiling of fbrick
  (0 0 1 -288) ((0.0078125 0 0.5) (0 -0.0078125 -1)) "textures/hell/cbrick2b" 0
  // top most horizontal of fbrick
  (-1 0 0 -480) ((0.0078125 0 0.5) (0 -0.0078125 -1)) "textures/hell/cbrick2b" 0
  // left most vertical of fbrick
  (0 -1 0 -576) ((0.0078125 0 0.5) (0 -0.0078125 -1)) "textures/hell/cbrick2b" 0
  // bottom most horizontal of fbrick
  (1 0 0 432) ((0.0078125 0 0.5) (0 -0.0078125 -1)) "textures/hell/cbrick2b" 0 0
  // right most vertical of fbrick
  (0 1 0 528) ((0.0078125 0 0.5) (0 -0.0078125 -1)) "textures/hell/cbrick2b" 0 0
}
```

- six planes which define a cuboid

How are these values calculated?

- step 1 obtain a list of vertices (list of corners) for your convex polyhedra
  - try this for a cuboid

```
[[1, 1, 1], [1, 2, 1], [1, 1, 2], [1, 2, 2],
 [2, 2, 2], [2, 2, 1], [2, 1, 2], [2, 1, 1]]
```

- step 2 calculate the six planes equations for this cube
  - one equation per face
- the plane equation is: $ax + by + cz + d = 0$

Finding the plane equation

- the plane equation can be found by taking the two vectors which represent the edge of the face
- obtain the cross product of the vectors

```
v0 = subVec ([1, 2, 1], [1, 1, 1])
v1 = subVec ([1, 1, 2], [1, 1, 1])
a, b, c = crossProduct (v0, v1)
```

Finding the plane equation

- `a, b, c` are actually the normal to the face
  - if you want the other normal you can perform

```
a, b, c = crossProduct (v1, v0)
```


Finding the plane equation

- must be careful and choose the correct one
  - note the right hand rule
  - first finger is v0
  - second finger is v1
  - thumb is the normal
- where

```
a, b, c = crossProduct (v0, v1)
```

- n = v0 × v1

Implementing the cross product in Python

```
#
#  crossProduct - return the cross product of a and b.
#                 a x b
#

def crossProduct (a, b):
    assert (len (a) == 3)
    assert (len (b) == 3)
    return [a[1] * b[2] - (b[1] * a[2]),
            a[2] * b[0] - (a[0] * b[2]),
            a[0] * b[1] - (a[1] * b[0])]
```

- the normal vector to the face contains the values
  - `a, b, c` required in the plane equation
- now we must calculate `d` (the closest distance the plane is from the origin)

Implementing the cross product in Python

- we want to find the nearest distance between the plane and origin
  - we also have a point on the plane, `p`, (one of the vertices of our polyhedra)

```
p = [...]
```

- $ax + by + cz + d = 0$

```
d = - (p[0] * a + b * p[1] + c * p[2])
```

Implementing the cross product in Python

- the values: `a`, `b`, `c` and `d` form part of the `brushdef3`
- ```
  brushDef3
  {
    ...
    (a b c d) ((0.0078125 0 0.5) (0 -0.0078125 -1)) "textur
    ...
  }
  ```
- the other values were explained in a previous lecture

Tutorial

- firstly fix the beams bug in `pen2map.py` which currently places a beam in the doorway
  - ie suppress the construction of a beam which will end in the doorway!
- secondly construct a cube from `brushdef3`'s by hand and add them into `$HOME/Sandpit/chisel/python/tiny.map`
- hint use an existing `tiny.map` and remove all `brushdef`'s
  - add a single `brushdef` containing your cuboid
- hint use Python and `chisel/python/chvec.py` to help you with the maths

Tutorial

- thirdly construct a sheared cuboid (from the side in 2D it would appear as a parallelogram)
  ```