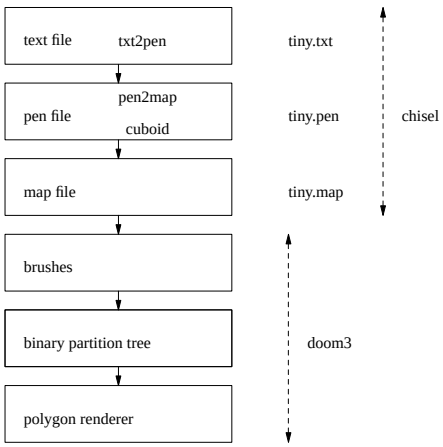
pen2map overview

- parses a pen file, creates internal data structures representing the pen map
  - it then iterates over the rooms and generates a doom3 map file
- conceptually the generation of the rooms is rather like virtualised lego (within chisel)
  - pen2map generates blocks and places these blocks into a world
  - it will attempt to join blocks together as long as this results in a bigger cuboid structure
- however the doom3 map uses planes and not blocks!

Construct the program in logical units

- Henry Legard proverb
- one way to achieve this is to layer the solution
  - divide and conquer
- consider our doom3 tools

Doom3 and chisel layering



Minimal box defined in the map format

```
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

The second plane

- is the ceiling in our example
- `(0 0 1 -288) ((0.0078125 0 0.5) (0 -0.0078125 -1)) "textures/hell/cbrick2b" 0 0 0`
- (0 0 1 -288)
  - vector (0, 0, 1) and the closest it reaches the origin is -288 units
  - this infinite plane will have the texture `textures/hell/cbrick2b` applied to it

Texture transformation matrix

- the texture uses the transformation matrix, T

$$T = \begin{bmatrix} 0.0078125 & 0 & 0.5 \\ 0 & -0.0078125 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

- general transformation matrix is:
- $$T = \begin{bmatrix} xscale\cos(\theta) & -yscale\sin(\theta) & translatex \\ xscale\sin(\theta) & yscale\cos(\theta) & translatey \\ 0 & 0 & 1 \end{bmatrix}$$

Each coordinate is transformed by

- $$T = \begin{bmatrix} xscale\cos(\theta) & -yscale\sin(\theta) & translatex \\ xscale\sin(\theta) & yscale\cos(\theta) & translatey \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
- and mapped into the image file at this new grid coordinate
- fortunately we conceptualise chisel as creating a variety of lego bricks (each is a cuboid)
- `pen2map.py` generates floor bricks, wall bricks and ceiling bricks

Conclusion

- layered software is an important concept which allows large systems to be built and it can hide complexity behind well defined interfaces
- cuboids are represented by brushes in the map
  - six planes define a brush

Tutorial

- finish off your automatic light code in `txt2pen.py`
- see if you can make the floor level vary
  - by lowering slightly every odd room number floor
  - leave the even room number floor alone
- need to examine and change `pen2map.py`