

Programming Proverbs

- 8. “Avoid side effects.”
- Henry F. Ledgard, “Programming Proverbs: Principles of Good Programming with Numerous Examples to Improve Programming Style and Proficiency”, (Hayden Computer Programming Series), Hayden Book Company, 1st edition, ISBN-13: 978-0810455221, December 1975.

Implementing labels in chisel/txt2pen

- it would be useful to introduce labels in maps
 - could be used for waypoints
 - or to create a taxonomy of objects

- wish to introduce the ability for a bot to get the
 - location of a label
 - doom3 units and also pen units
 - whether a label exists
 - a list of all labels

Implementing labels in chisel/txt2pen

- we might expect, eventually, to enhance the bot API by providing methods inside `botlib.py`. For example:

```
l = botlib.get_all_labels ()
...
doom_loc = botlib.get_d3_loc (label_name)
...
pen_loc = botlib.get_pen_loc (label_name)
...
if botlib.has_label (label_name):
    ...
```

- some of these can be implemented in Python on the client
 - others will interact with the server
 - all can be cached

Implementing labels in chisel/txt2pen

- a thought exercise for the reader might be to consider how labels might be automatically generated
 - for example every room number might automatically generate a label
 - each player and bot might also be known by a label
 - every pickup can also be known by a label

- these notes will only address how labels can be manually entered on maps

Implementing labels in chisel/txt2pen

`$HOME/Sandpit/chisel/map/label.txt`

```
define l room 1
define s worldspawn
define o monster monster_demon_imp
define n monster monster_demon_hellknight
define i light
define a ammo ammo_shells_large 16
define l label the_ammo_loc [a]

#####
# l      i                                     #
#                                             #
# s                                     l      #
#                                             #
#                                             #
#####
```

Implementing labels in chisel/txt2pen

- this work is divided into 3 components
 - firstly change `$HOME/Sandpit/chisel/python/txt2pen.py` to generate labels inside the `.pen` files
 - secondly change `$HOME/Sandpit/chisel/python/pen2map.py` to place labels inside doom3 map files
 - thirdly change `$HOME/Sandpit/git-doom3/pybot-dhewm3/python-bot/botaa.py`

- these notes address the first part only

Implementing labels in chisel/txt2pen

- making the changes in these notes in Sandpit/chisel/python/txt2pen.py **will** break the existing doom3 python bot and also break pen2map.py
- so, we will make these changes in a different copy of chisel (stored in Sandpit/labels)

```
$ cd  
$ cd Sandpit  
$ mkdir -p labels  
$ cd labels  
$ git clone https://github.com/gaiusm/chisel
```

Changes to Sandpit/labels/chisel/python/txt2pen.py

- add a new field to the `roomInfo` class

- `$HOME/Sandpit/labels/chisel/python/txt2pen.py`

```
class roomInfo:
    def __init__ (self, w, d):
        ...
        self.defaultColour = {}
        self.defaultTexture = {}
        self.sounds = []
        self.labels = []
```


Changes to Sandpit/labels/chisel/python/txt2pen.py

- update the reserved words:

- `$HOME/Sandpit/labels/chisel/python/txt2pen.py`

```
reservedKeywords = ['ammo', 'ceiling', 'colour', 'default',  
                    'floor', 'label', 'light', 'looping',  
                    'mid', 'monster',  
                    'worldspawn',  
                    'room', 'sound',
```

Changes to Sandpit/labels/chisel/python/txt2pen.py

- add the label class

- `$HOME/Sandpit/labels/chisel/python/txt2pen.py`

```
class label:
    def __init__ (self, pos, label_desc):
        self.pos = pos
        self.label_desc = label_desc
    def write (self, f):
        f.write (" LABEL AT ")
        printCoord (self.pos, f)
        f.write (" %s\n" % self.label_desc)
        return f
```

Changes to Sandpit/labels/chisel/python/txt2pen.py

- add parseLabel

- `$HOME/Sandpit/labels/chisel/python/txt2pen.py`

```
#  
# parseLabel := 'label' filename =:  
#  
  
def parseLabel (room, x, y):  
    desc = expectString (room, x, y, 'a string after the label keyword')  
    l = label ([x, y], desc)  
    rooms[room].labels += [l]
```

Changes to Sandpit/labels/chisel/python/txt2pen.py

- add parseLabelSpawn

- `$HOME/Sandpit/labels/chisel/python/txt2pen.py`

```
#  
# parseLabelSpawn := 'label' string =:  
#  
  
def parseLabelSpawn (room, x, y):  
    if expecting (['label']):  
        expect ('label', room, x, y)  
        parseLabel (room, x, y)  
        return True  
    return False
```

Changes to Sandpit/labels/chisel/python/txt2pen.py

- we need to make a call to `parseLabelSpawn` from within the function `ebnf`

- `$HOME/Sandpit/labels/chisel/python/txt2pen.py`

```
def ebnf (room, x, y):  
    ...  
    pass  
    elif parseSoundSpawn (room, x, y):  
        pass  
    elif parseLabelSpawn (room, x, y):  
        pass  
    else:  
        w = tokens.split () [0]  
        error ("unexpected token " + w + " in room " +
```

Changes to Sandpit/labels/chisel/python/txt2pen.py

- finally we need to add a method to print labels (and make sure it is called)

- `$HOME/Sandpit/labels/chisel/python/txt2pen.py`

```
def printLabels (labels, o):  
    if labels != []:  
        for l in labels:  
            o = l.write (o)  
    return o
```

Changes to Sandpit/labels/chisel/python/txt2pen.py

■ `$HOME/Sandpit/labels/chisel/python/txt2pen.py`

```
def printRoom (r, o):  
    o = printSpawnPlayer (rooms[r].worldspawn, o)  
    o = printInside (rooms[r].inside, o)  
    o = printSounds (rooms[r].sounds, o)  
    o = printLabels (rooms[r].labels, o)  
    o.write ("END\n\n")  
    return o
```