

John Romero Programming Proverbs

- 9. “Encapsulate functionality to ensure design consistency. This minimizes mistakes and saves design time.”
- John Romero, “The Early Days of Id Software - John Romero @ WeAreDevelopers Conference 2017”

Implementing labels in chisel/pen2map

- you will need to back port the changes you made last week into your working chisel `$HOME/Sandpit/chisel`
- an easy way to do this is to use `diff` and `patch`
 - hint read `man diff` and `man patch`

```
$ cd $HOME/Sandpit/labels/chisel
$ git diff
$ git diff > $HOME/Sandpit/labels/label-diffs
$ cd $HOME/Sandpit/chisel
$ patch -p1 < $HOME/Sandpit/labels/label-diffs
```

Implementing labels in chisel/pen2map

- the command `git diff` generates a list of differences between each file and the last git clone
- the command `patch` tries to apply these differences to the current directory
 - this is used a lot in industry and it is a very efficient way to distribute source code changes
 - occasionally you need to manual apply a patch which fails
 - normally `patch` will complain and store a `.rej` file
- these notes are better read using the pdf version

Implementing labels in chisel/pen2map

`$HOME/Sandpit/chisel/maps/label3.txt`

```
define 1 room 1
define s worldspawn
define i light
define a ammo ammo_shells_large 16
define l label the_ammo_loc [a]
define P monster python_doommarine_mp

#####
# 1                                     #
#           P                         #
# s                                     #
#           1                         #
#                                     #
#####
```

Implementing labels in chisel/pen2map

- given the map source file above can be transformed into a pen map using
- ```
$ cd $HOME/Sandpit/chisel/python
$ python3 txt2pen.py ../maps/label13.txt
```
- if you want autolights you need to add the `-l` option

# Implementing labels in chisel/pen2map

```
ROOM 1
 WALL
 1 7 36 7
 36 7 36 1
 36 1 1 1
 1 1 1 7
 MONSTER python_doommarine_mp AT 12 5
 AMMO ammo_shells_large AMOUNT 16 AT 24 4
 SPAWN PLAYER AT 4 4
 INSIDE AT 4 5
 LABEL AT 24 4 the_ammo_loc
END

END.
```

## Adding labels to chisel/pen2map

- keep the comment grammer up to date

■ [python/pen2map.py](#)

```
roomDesc := "ROOM" integer
 { doorDesc | wallDesc | treasureDesc | ammoDesc |
 lightDesc | insideDesc | weaponDesc | monsterDesc |
 spawnDesc | defaultDesc | soundDesc | labelDesc } =:
```

- which states a room starts with a keyword ROOM followed by an integer followed by a description
  - one of which is the labelDesc

## Adding labels to chisel/pen2map



python/pen2map.py

```
labelDesc := 'LABEL' 'AT' posDesc string =:
```



the grammar comment is at the top of the source file in a multi line comment



## Adding labels to chisel/pen2map

python/pen2map.py

```
class roomInfo:
 def __init__ (self, r, w, d):
 self.walls = orderWalls (r, w)
 ...
 self.floorLevel = None
 self.inside = None
 self.defaultColours = {}
 self.defaultTextures = {}
 self.sounds = []
 self.labels = []
```

## Adding labels to chisel/pen2map



python/pen2map.py

```
def addSound (self, s, pos):
 self.sounds += [[s, pos]]
def addLabel (self, label, pos):
 self.labels += [[label, pos]]
```

## Adding labels to chisel/pen2map

python/pen2map.py

```
def roomDesc ():
 global curRoom, curInteger, curRoomNo, verbose
 if expecting (['ROOM']):
 expect ("ROOM")
 if integer ():
 curRoomNo = curInteger
 curRoom = newRoom (curRoomNo)
 if debugging:
 print("roomDesc", curRoomNo)
 while expecting (['DOOR', 'WALL', 'TREASURE', 'AMMO', 'WEAPON', 'LIGHT', '1']):
 if expecting (['DOOR']):
 doorDesc ()
 elif expecting (['WALL']):
 wallDesc ()
```

## Adding labels to chisel/pen2map

python/pen2map.py

```
...
elif expecting (['WEAPON']):
 weaponDesc ()
elif expecting (['LABEL']):
 labelDesc ()
elif expecting (['LIGHT']):
 lightDesc ()
elif expecting (['INSIDE']):
 insideDesc ()
elif expecting (['MONSTER']):
 monsterDesc ()
...
expect ('END')
return True
else:
 errorLine ('expecting an integer after ROOM')
return False
```

## Adding labels to chisel/pen2map

python/pen2map.py

```
#
labelDesc := 'LABEL' 'AT' posDesc string =:
#

def labelDesc ():
 expect ('LABEL')
 expect ('AT')
 if posDesc ():
 label = get ()
 curRoom.addLabel (label, curPos)
 return True
 else:
 errorLine ('expecting a position for a label')
 return False
```

## Adding labels to chisel/pen2map

python/pen2map.py

```
def generateWeapons (o, e):
 n = 1
 for r in list(rooms.keys ()):
 if debugging:
 print(rooms[r].weapons)
 for weapon_kind, xy in rooms[r].weapons:
 o.write ("// entity " + str (e) + '\n')
 o.write ("{\n")
 o.write (' "inv_item" "4"\n')
 o.write (' "classname" "' + weapon_kind + '"\n')
 o.write (' "name" "' + weapon_kind + '"\n')
 o.write (' "origin" "')
 xyz = toIntList (xy) + [-invSpawnHeight]
 xyz = subVec (xyz, [minx, miny, getFloorLevel (r)])
 v = midReposition (xyz)
 o.write ('%f %f %f\n' % (v[0], v[1], v[2]))
 o.write ("}\n")
 n += 1
 e += 1
 return o, e
```

## Adding labels to chisel/pen2map

python/pen2map.py

```
def generateLabels (o, e):
 n = 1
 for r in list(rooms.keys ()):
 if debugging:
 print (rooms[r].labels)
 for label_desc, xy in rooms[r].labels:
 o.write ("// entity " + str (e) + '\n')
 o.write ("{\n")
 o.write (' "classname" "item_default"\n')
 o.write (' "label" "' + label_desc + '"\n')
 o.write (' "origin" "')
 xyz = toIntList (xy) + [-invSpawnHeight]
 xyz = subVec (xyz, [minx, miny, getFloorLevel (r)])
 v = midReposition (xyz)
 o.write ('%f %f %f"\n' % (v[0], v[1], v[2]))
 o.write ("}\n")
 n += 1
 e += 1
 return o, e
```

## Adding labels to chisel/pen2map



python/pen2map.py

```
def generateMap (o):
 o, e = generateLights (o, e)
 o, e = generateAmmo (o, e)
 o, e = generateSounds (o, e)
 o, e = generateWeapons (o, e)
 o, e = generateLabels (o, e)
 if statistics:
 print("Total rooms =", len (list(rooms.keys ())))
 print("Total cuboids =", len (list(cuboids.keys ())))
```



## Testing the tool chain

```
$ cd $HOME/Sandpit/chisel/python
$ python3 txt2pen.py -l ../maps/label13.txt
$ python3 txt2pen.py -l -o tiny.pen ../maps/label13.txt
$ cat tiny.pen
$ PYTHONPATH=student python3 pen2map.py tiny.pen
$ PYTHONPATH=student python3 pen2map.py -o tiny.map tiny.pen
```

if this works then you can simplify these commands to:

```
$ cd $HOME/Sandpit/chisel/python
$./developer-txt2map ../maps/label13.txt
```

## Testing the tool chain

- now download a new copy of doom3 and try out the new bot code
  - you will need to remove the current pybot-dhewm3 code (or move it out of the way)

- ```
$ ssh mcgreg
$ cd $HOME/Sandpit
$ git clone https://github.com/gaiusm/pybot-dhewm3
$ exit
```

- now recompile doom3

Testing the tool chain

- start the python bot

- ```
$ cd $HOME/Sandpit/git-doom3/pybot-dhewm3/python-bot
$ python3 python_doommarine.py 0
```

- examine the code in `$HOME/Sandpit/git-doom3/pybot-dhewm3/python-bot/python_doommarine.py`
- look at the code which obtains the label doom3 coordinate
  - the python bot will briefly come to life and then print out the label position