CS3S665: Game Engine Design Coursework 1

Khalid Ali (15005070)

December 2, 2019

Contents

1 Introduction

1.1 The tools

Chisel is a two-stage Python-based map format converter. The first stage or part of the tool is **txt2pen**, which converts an ASCII character map into a Penguin Tower map. The second stage is **pen2map**, which converts a Penguin Tower map into a Doom 3 map. Pybot is a Python/C++ bot for Doom 3. In its present state, it can only track and move towards the player.

1.2 Starting state

When we started, Chisel was lacking automatic lighting, weapons generation, and roof beams were being calculated too low. There was also gaps in commenting. Pybot was missing critical functionality such as returning ammo, returning health, allowing reloading of weapons, selecting weapons, and stepping up/down.

1.3 What we have to do

We were tasked with exploring and extending both Chisel and Pybot. Chisel requires implementation of various tutorial tasks, as well as my own modifications that could enhance its operation. Pybot was similar, requiring me to complete tutorials and implement some intelligence into the bot.

2 Extending Chisel

My extensions of Chisel was done in two parts; following changes suggested by the tutorials and then implementing some of my own. The tutorials asked and the assignment brief suggested these among the changes to be made:

- Introducing lighting
- Improving how roof beams are implemented
- Modifying how floor steps are implemented
- Testing weapon drops

2.1 Automatic lighting (txt2pen)

Introducing automatic lighting was requested early in tutorials, and involves replicating and modifying the **scanRoom()** method and testing it by calling **txt2pen** with the **-l** option. The method should also only work if the rooms have no user-defined lights.

2.2 Improving roof beams (pen2map)

Improving how beams are implemented is suggested as part of the assessment. During in game play, it is obvious that the beams upon the roof are too low and break the cleanness of the scene.

Upon inspection, I noticed that the height of the beams are controlled by calling methods **makeBeamX()** and **makeBeamY()** in **generateBeams()**.

```
makeBeamX (x, y0, y1, rooms[r].floorLevel+minCeilingHeight-1)
makeBeamY (x, y0, y1, rooms[r].floorLevel+minCeilingHeight-1)
```

I am unsure why "-1" is used at the end of the calls, so both instances were removed and the beams now look cleaner.

2.3 Modifying floor levels (pen2map)

I decided to explore how floor levels and steps were being processed. The tutorial suggestion of dropping the floor height when the room number is odd brought about a puzzling bug.

Removed:

```
self.floorLevel = None
```

Added:

```
if int (r) % 2 == 0:
    self.floorLevel = 0
else:
    self.floorLevel = -0.25
```

This resulted in the steps generated being far too big compared to the change in floor level. I was unsure if whether the amount the floor level is changed was wrong, or whether the stairs were just being generated far too big.

Given time constraints, I opted to visually alter the steps to match the floor level. I modified two global variables at the top of **pen2map**. I significantly lowered the size of each step and increased the number of steps to provide a much more visually-consistent step. One thing to note is that the steps overall look small, which I am unsure as to whether that should be the end result.

The variables I changed:

```
floorStep       = 0.05
noSteps         = 5
```


0
81
UNIDENTIFIED

3 Pybot

3.1 Context

Pybot is a layered approach to controlling a Doom 3 marine character. The primary way we can interface is through Python, to which we should note two Python mechanisms; **botbasic** and **botcache**.

Botbasic provides the main interface to specify logic, providing basic commands such as move and fire. Botcache has the same interface as botbasic, but it will cache the results of interfacing with the Doom application so that other methods can query the values without overhead. To extend from this, we can create a new Python script such as **python_doommarine.py** to implement our own methods and alter the Pybot's execution loop.

3.2 Jumping & crouching

The ability to jump and crouch were specified within tutorials, which required that we implement stepping up functionality based on the already provided step forward and step vector. Once step up was implemented, it was just a matter of specifying the direction of velocity and distance of movement for both via **stepup()**.

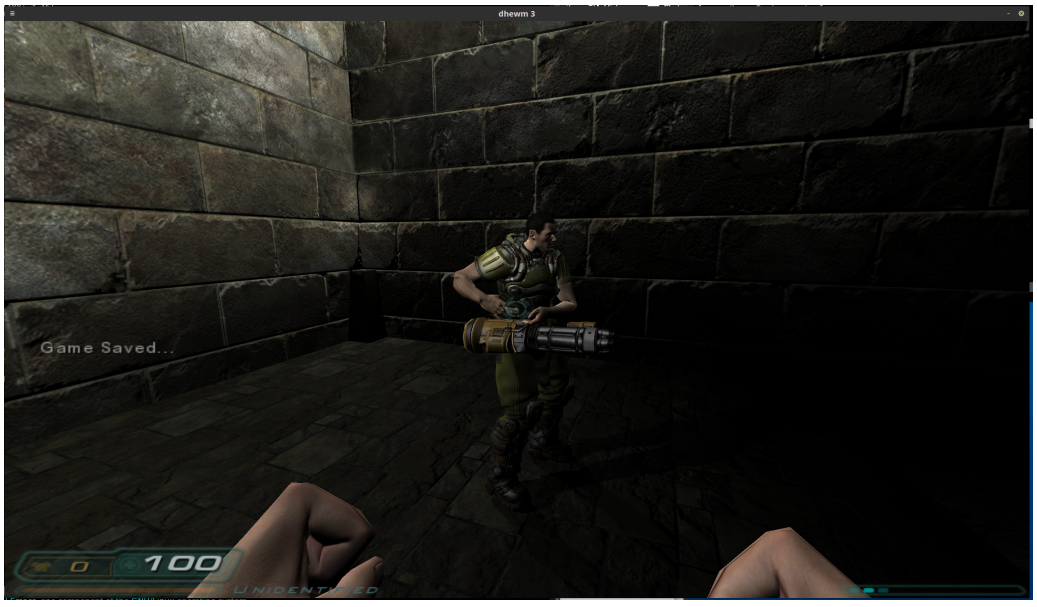
```
def jump (b):
    b.reset ()
    b.stepup (100, 4*12)
```


```
def crouch (b):
    b.reset ()
    b.stepup (-5, 3*12)
    b.select (['move'])
```

3.3 Changing Weapon

Change weapon works fine, although weapons other than the pistol have no ammo so the weapon will change back to a pistol once the bot attempts to fire.

```
b.changeWeapon (4)
```



3.4 Ammo

I noticed that the amount of ammo the bot currently has was not being returned by methods such as start_firing (), so digging into the Player class code revealed that the ammo values stored in inventory were a single index ahead of **currentWeapon**. So adding 1 temporarily fixed this (I ran out of time before finding a more elegant better solution)

```
return inventory.ammo[currentWeapon + 1];
```

4 In retrospect

4.1 Using the tools

With problems I have created aside, I believe both tools have their merit. Chisel worked well from the start, and I was able to compile a wide range of Doom 3 maps. It was fun trying to experiment with my own unique room designs, especially after completing my extensions! I have no complaints the usage process over that perhaps more custom error messages in contrast to relying on just Python tracebacks for debugging. Pybot required a lot more to be done before I got a result I can see, however likewise it was funny to see the initial quirks such as causing the bot to endlessly crouching. Debugging, however, was initially frustrating since gaps in implementations or misscommunication between bot and server would result in the bot simply declaring it was dead. The exploration to resolve such errors, however, increased my understanding on what was going wrong so future issues could be more quickly and effortlessly resolved.

4.2 Broken modification

At present, **reload_weapon()** does not work in my solution. It will cause the bot to kill itself as soon as it is called, something I believe is due to my own mistake. It is the only major and blantant broken feature, that if fixed I could build a bot with usable function.

4.3 Code commenting

Commenting was one thing I was hoping to look before starting the assignment. I noticed there was a scope for improvement in the comments since there were some gaps some of the core Python files, so I began filling them in (see **pen2map.py** in Appendix). Unfortunately, I had to cut this short due to dwindling time. If could have done it, the process of exploring and analysing methods would have increased my ability to understand how I could better extend both Chisel and Pybot. I admit that tutorial time could have given me the opportunity, which I missed and payed for.

4.4 Code reduction

As with commenting, I believe their was scope for cutting down the amount of Python code. If I had more time, I would have liked to explore if I could perhaps refactor and simplify the code. Areas I would have liked to have looked at would be the amount of Python files (merging botbasic and botcache into a new solution) and seeing if large/similar Chisel methods such as **scanRoom()** and **introduceLights()** could be simplified and merged.

5 Appendix

5.1 Test Map: modified spiral.txt

```
define 1 room 1
define 2 room 2
define 3 room 3
define O monster monster_demon_imp
define P monster python_doommarine_mp
define S worldspawn
define i light

##########################
#i1    i#i2          i#
#       .             #########
#       .  iiiiiiiiii        #
#       .             #########
#i  S   i#i           i#
#........################
#i3          i#
#             #
#         P   #
#             #
#i           i#
################
```

5.2 GIT DIFF: Chisel

```
diff --git a/python/pen2map.py b/python/pen2map.py
index 0d2514d..416ed82 100644
--- a/python/pen2map.py
+++ b/python/pen2map.py
@@ -143,8 +143,8 @@ lightFloorHeight = 0.0625 # 3 inches
 lightFloorHeight = 0.125    # 6 inches
 lightFloorHeight = 0.125    # 6 inches
 lightCeilingHeight = minCeilingHeight - 1.5
-floorStep        = 0.25     # 1 foot
-noSteps          = 4        # how many steps
+floorStep        = 0.05     # changed for smooth chairs
+noSteps          = 5        # how many steps


 def mycut (l, i):
@@ -298,7 +298,10 @@ class roomInfo:
         self.ammo = []
         self.lights = []
         self.worldspawn = []
-        self.floorLevel = None
+        if int (r) % 2 == 0:
+            self.floorLevel = 0
+        else:
+            self.floorLevel = -0.25
         self.inside = None
         self.defaultColours = {}
         self.sounds = []
@@ -2273,7 +2276,7 @@ def generateBeams (r, e):
                 print w0, w1
             for x in range (w0[0][0]+1, w0[1][0], 2):
                 if (not onDoor (r, x, y0)) and (not onDoor (r, x, y1)):
-                    makeBeamX (x, y0, y1, rooms[r].floorLevel+minCeilingHeight-1)
+                    makeBeamX (x, y0, y1, rooms[r].floorLevel+minCeilingHeight-0.5)
             for x in range (w0[0][0]+2, w0[1][0], 2):
                 if (not onDoor (r, x, y0)) and (not onDoor (r, x, y1)):
                     makeCandleX (r, x, y0, y1, rooms[r].floorLevel+minCeilingHeight-2)
@@ -2292,7 +2295,7 @@ def generateBeams (r, e):
                 print w0, w1
             for y in range (w0[0][1]+1, w0[1][1], 2):
                 if (not onDoor (r, x0, y)) and (not onDoor (r, x1, y)):
-                    makeBeamY (x0, x1, y, rooms[r].floorLevel+minCeilingHeight-1)
+                    makeBeamY (x0, x1, y, rooms[r].floorLevel+minCeilingHeight-0.5)
             for y in range (w0[0][1]+2, w0[1][1], 2):
                 if (not onDoor (r, x0, y)) and (not onDoor (r, x1, y)):
                     makeCandleY (r, x0, x1, y, rooms[r].floorLevel+minCeilingHeight-2)
@@ -2640,22 +2643,36 @@ def generateMonsters (o, e):
             e += 1
     return o, e

+def generateWeapons (o, e):
+    n = 1
+    for r in rooms.keys():
+        print rooms[r].weapons
+        for t, p in rooms[r].weapons:
```

```
+           o.write ("// entity " + str (e) + '\n')
+           o.write ("{\n")
+           o.write ('   "inv_item" "4"\n')
+           o.write ('   "classname" "' + "rocket_launcher" + '"\n')
+           o.write ('   "origin" "')
+           o = writeMidPos (o, p)
+           o.write (' ' + str (toInches (getFloorLevel (r) + invSpawnHeight - minz + 1.0)) +
    '"\n')
+           o.write ("}\n")
+           n += 1
+           e += 1
+   return o, e

 def generateAmmo (o, e):
     n = 1
     for r in rooms.keys():
         if debugging:
             print rooms[r].ammo
-       for ammo_kind, a, xy in rooms[r].ammo:
+       for t, a, p in rooms[r].ammo:
             o.write ("// entity " + str (e) + '\n')
             o.write ("{\n")
-           o.write ('   "inv_item" "4"\n')
-           o.write ('   "classname" "' + ammo_kind + '"\n')
-           o.write ('   "name" "' + ammo_kind + '_' + str (n) + '"\n')
+           o.write ('   "classname" "' + t + '"\n')
+           o.write ('   "name" "' + t + '_' + str (n) + '"\n')
             o.write ('   "origin" "')
-           xyz = xy + [getFloorLevel (r) + invSpawnHeight]
-           v = pen2MidPos (xyz)
-           o.write ('%f %f %f"\n' % (-v[0], -v[1], v[2]))
+           o = writeMidPos (o, p)
+           o.write (' ' + str (toInches (getFloorLevel (r) + invSpawnHeight - minz + 1.0)) +
    '"\n')
             o.write ("}\n")
             n += 1
             e += 1
@@ -2695,18 +2712,22 @@ def generateSounds (o, e):
             e += 1
     return o, e

-
-def assignFloorLevel (f):
+# Assigns floor level
+def assignFloorLevel():
     global rooms
+    print (rooms)
     for r in rooms.keys():
-       rooms[r].floorLevel = f
+       if int (r) % 2 == 0:
+            self.floorLevel = 0
+       else:
+           self.floorLevel = -0.25


 def generateMap (o):
```

```
     if genSteps:
         calcFloorLevel ()
     else:
-        assignFloorLevel (0)
+        assignFloorLevel()
     o.write ("// automatically created from: " + inputFile + "\n")
     o       = generateVersion (o)
     o, e, b = generateEntities (o)
@@ -2715,6 +2736,7 @@ def generateMap (o):
     o, e    = generateMonsters (o, e)
     o, e    = generateLights (o, e)
     o, e    = generateAmmo (o, e)
+    o, e    = generateWeapons (o, e)
     o, e    = generateSounds (o, e)
     if statistics:
         print "Total rooms =", len (rooms.keys ())
diff --git a/python/txt2pen.py b/python/txt2pen.py
index 58d19a4..8e5f723 100644
--- a/python/txt2pen.py
+++ b/python/txt2pen.py
@@ -78,6 +78,9 @@ def initFloor (x, y, value):
         row = [value] * (x+1)
         floor += [row]

+#
+#   roomInfo - contains properties of parsed rooms
+#

 class roomInfo:
     def __init__ (self, w, d):
@@ -132,6 +135,9 @@ def debugf (format, *args):
     if debugging:
         print str (format) % args,

+#
+#   usage - prints parametre help on how to use txt2pen
+#

 def usage (code):
     print "Usage: txt2pen [-dhlvV] [-f frequency] [-o outputfile] inputfile"
@@ -146,7 +152,7 @@ def usage (code):


 #
-#  handleOptions -
+#  handleOptions - handles accepting parametres given to txt2pen
 #

 def handleOptions ():
@@ -293,18 +299,34 @@ def getListOfRooms (mapGrid, start, i):
     return listOfRooms, pos


+#
+#   isWall - returns whether entity is wall
+#
```

```
+
 def isWall (pos, grid):
     return grid[pos[1]][pos[0]] == '#'


+#
+#   isDoor - returns whether entity is door
+#
+
 def isDoor (pos, grid):
     return (grid[pos[1]][pos[0]] == '-') or (grid[pos[1]][pos[0]] == '|') or
         (grid[pos[1]][pos[0]] == '.')


+#
+#   isPlane - returns whether entity is plane
+#
+
 def isPlane (pos, grid):
     return isWall (pos, grid) or isDoor (pos, grid)


+#
+#   addVec - returns product of vectors pos and vec
+#
+
 def addVec (pos, vec):
     return [pos[0]+vec[0], pos[1]+vec[1]]

@@ -341,6 +363,10 @@ def addWall (walls, start, current):
     return walls, current


+#
+#   lookingLeft -
+#
+
 def lookingLeft (pos, left, grid, s):
     if debugging:
         print pos, left, s
@@ -363,6 +389,11 @@ def lookingLeft (pos, left, grid, s):
 def mystop ():
     pass

+
+#
+#  scanRooms - scans content of a room
+#
+
 def scanRoom (topleft, p, mapGrid, walls, doors):
     global debuging
     if debugging:
@@ -445,6 +476,7 @@ def scanRoom (topleft, p, mapGrid, walls, doors):
         else:
             error ("scanning room at %s has gone wrong, maybe the room is too small\n", p)
```

```
+
 #
 #  checkLight - add a mid light if lightCount == lightFrequency
 #
@@ -460,9 +492,75 @@ def checkLight (p, l, lightCount):
     return l, lightCount


+#
+#  introduceLights - computes specified (not automatic) lighting
+#
+
 def introduceLights (topleft, p, mapGrid, walls, doors):
-    # your code goes here
-    return [] # and replaces this line
+    global debuging
+    s = p
+    a = addVec (p, [-1, -1])
+    # 0 up, 1 right, 2 down, 3 left
+    d = 1
+    leftVec = [[-1, 0], [0, -1], [1, 0], [0, 1]]
+    forwardVec = [[0, -1], [1, 0], [0, 1], [-1, 0]]
+    lightCount = 0
+    lights = []
+    doorStartPoint = None
+    doorEndPoint = None
+    needToAvoidDoor = False
+    while True:
+        if debugging:
+        if (doorStartPoint == None) and lookingLeft (p, leftVec[d], mapGrid, '. '):
+            if debugging:
+            doorStartPoint = addVec (p, leftVec[d])
+            doorEndPoint = doorStartPoint
+            needToAvoidDoor = True
+        if lookingLeft (addVec (p, forwardVec[d]), leftVec[d], mapGrid, '. '):
+            if debugging:
+            if doorStartPoint == None:
+                doorStartPoint = addVec (addVec (p, forwardVec[d]), leftVec[d])
+            doorEndPoint = addVec (addVec (p, forwardVec[d]), leftVec[d])
+            needToAvoidDoor = True
+        else:
+            if doorEndPoint != None:
+                doorStartPoint = None
+                doorEndPoint = None
+                needToAvoidDoor = True
+        if lookingLeft (addVec (p, forwardVec[d]), leftVec[d], mapGrid, 'x '):
+            if needToAvoidDoor:
+                li = light ()
+                li.settype ('FLOOR')
+                lights += [p + [li]]
+            else:
+                lights, lightCount = checkLight (p, lights, lightCount)
+            needToAvoidDoor = False
+            p = addVec (p, forwardVec[d])
+        elif lookingLeft (addVec (p, forwardVec[d]), leftVec[d], mapGrid, 'x.'):
+            if debugging:
```

```
+           doorStartPoint = None
+           doorEndPoint = None
+           d = (d + 1) % 4
+           if s == p:
+               return lights
+       elif lookingLeft (addVec (p, forwardVec[d]), leftVec[d], mapGrid, 'xx'):
+           if debugging:
+           doorStartPoint = None
+           doorEndPoint = None
+           d = (d + 1) % 4
+           if s == p:
+               return lights
+       elif lookingLeft (addVec (p, forwardVec[d]), leftVec[d], mapGrid, ' '):
+           if debugging:
+           p = addVec (p, forwardVec[d])
+           d = (d + 3) % 4
+           if s == p:
+               return lights
+
+
+#
+#  printCoord - writes out given coordinate
+#

 def printCoord (c, o):
     global maxy
@@ -470,6 +568,10 @@ def printCoord (c, o):
     return o


+#
+#  printMonsters - write out the monster spawn position and characteristics.
+#
+
 def printMonsters (mlist, o):
     if mlist != []:
         for kind, pos in mlist:
@@ -479,6 +581,10 @@ def printMonsters (mlist, o):
     return o


+#
+#  printSpawnPlayer - write out the player spawn position.
+#
+
 def printSpawnPlayer (m, o):
     if m != []:
         for pos in m:
@@ -488,6 +594,10 @@ def printSpawnPlayer (m, o):
     return o


+#
+#  printInside - write out the inside data.
+#
+
```

```
 def printInside (inside, o):
     if inside != None:
         o.write ("  INSIDE AT ")
@@ -496,6 +606,10 @@ def printInside (inside, o):
     return o


+#
+#  printSounds - write out the sound position and characteristics.
+#
+
 def printSounds (sounds, o):
     if sounds != []:
         for s in sounds:
@@ -503,6 +617,10 @@ def printSounds (sounds, o):
     return o


+#
+#  printAmmo - write out the ammo position and characteristics.
+#
+
 def printAmmo (m, o):
     if m != []:
         for name, amount, pos in m:
@@ -512,6 +630,11 @@ def printAmmo (m, o):
     return o


+#
+#  printWeapons - write out the weapon position and characteristics.
+#
+
+
 def printWeapons (w, o):
     if w != []:
         for name, pos in w:
@@ -957,7 +1080,6 @@ def parseSound (room, x, y):
             s.setWait (n)
     rooms[room].sounds += [s]

-
 #
 #  parseEntity -
 #
@@ -998,8 +1120,7 @@ def parseEntity (room, x, y):
         parseSound (room, x, y)
     else:
         w = tokens.split ()[0]
-        error ("unexpected token " + w + " in room " +
-               str (room) + " at " + str (x) + " " + str (y))
+        error ("unexpected token " + w + " in room " + str (room) + " at " + str (x) + " " +
    str (y))


 #
```

5.3 GIT DIFF: Pybot (Python)

```
diff --git a/python-bot/botbasic.py b/python-bot/botbasic.py
index 40aa4b4..cd3e3ca 100644
--- a/python-bot/botbasic.py
+++ b/python-bot/botbasic.py
@@ -312,7 +312,21 @@ class basic:

     def back (self, vel, dist):
         return self.forward (-vel, dist)
+
+    #
+    #  stepup -
+    #
+

+    def stepup (self, velocity, dist):
+        l = "step_up %d %d\n" % (velocity, dist)
+        if debug_protocol:
+            print "requesting a", l
+        self.s.send (l)
+        l = self.getLine ()
+        if debug_protocol:
+            print "doom returned", l
+        return int (l)

     #
     #  stepvec - step along a vector at velocity [velforward, velright] for a, dist.
@@ -378,7 +392,7 @@ class basic:
     #               It returns the amount of ammo left.
     #

-    def startFiring (self):
+    def start_firing (self):
         if debug_protocol:
             print "requesting to fire weapon"
         self.s.send ("start_firing\n")
@@ -393,7 +407,7 @@ class basic:
     #               It returns the amount of ammo left.
     #

-    def stopFiring (self):
+    def stop_firing (self):
         if debug_protocol:
             print "requesting to stop firing weapon"
         self.s.send ("stop_firing\n")
@@ -404,11 +418,11 @@ class basic:


     #
-    #  reloadWeapon - reload the current weapon
+    #  reload_weapon - reload the current weapon
     #                It returns the amount of ammo left.
     #

-    def reloadWeapon (self):
```

```
+    def reload_weapon (self):
         if debug_protocol:
             print "requesting to reload weapon"
         self.s.send ("reload_weapon\n")
@@ -425,7 +439,7 @@ class basic:
     #               of ammo left for the weapon
     #               >= 0 if the weapon exists
     #               or -1 if the weapon is not in
-    #               the bots inventory.
+    #               the bot's inventory.
     #

     def changeWeapon (self, n):
diff --git a/python-bot/botcache.py b/python-bot/botcache.py
index 59a5220..e4c9e57 100644
--- a/python-bot/botcache.py
+++ b/python-bot/botcache.py
@@ -170,7 +170,25 @@ class cache:

     def stepvec (self, velforward, velright, dist):
         self.delpos (self.me ())
-        return self._basic.stepvec (velforward, velright, dist)
+        return self._basic.stepvec (velforward, velright, dist)
+
+     #
+     #  stepup -
+     #
+
+
+    def stepup (self, velup, dist):
+        self.delpos (self.me ())
+        return self._basic.stepup (velup, dist)
+
+    #
+    #  reload_weapon - reload the current weapon
+    #                  It returns the amount of ammo left.
+    #
+
+
+    def reload_weapon (self):
+        return self._basic.reload_weapon ()

     #
     #  sync - wait for any event to occur.
@@ -218,6 +236,14 @@ class cache:
         return self._basic.reload_weapon ()

     #
+    #  changeWeapon - changes the weapon
+    #                  It returns the amount of ammo left.
+    #
+
+    def changeWeapon (self, n):
+        return self._basic.changeWeapon (n)
+
+    #
```

```
     #  ammo - returns the amount of ammo for the current weapon.
     #

diff --git a/python-bot/botlib.py b/python-bot/botlib.py
index 218a156..34e1c6a 100644
--- a/python-bot/botlib.py
+++ b/python-bot/botlib.py
@@ -145,6 +145,14 @@ class bot:
     def right (self, vel, dist):
         return self._cache.right (vel, dist)

+     #
+     #  stepup - makes the bot jump or crouch.
+     #
+
+    def stepup (self, velup, dist):
+        return self._cache.stepup (velup, dist)
+
     #
     #  atod3 - convert a penguin tower map angle into the doom3 angle.
     #
@@ -169,6 +177,44 @@ class bot:
         return self._cache.select (1)

     #
+    #   - reload the current weapon
+    #                It returns the amount of ammo left.
+    #
+
+    def start_firing (self):
+        return self._cache.start_firing ()
+
+    #
+    #   - reload the current weapon
+    #                It returns the amount of ammo left.
+    #
+
+    def stop_firing (self):
+        return self._cache.stop_firing ()
+
+
+    #
+    #  reload_weapon - reload the current weapon
+    #                It returns the amount of ammo left.
+    #
+
+    def reload_weapon (self):
+        return self._cache.reload_weapon ()
+
+
+    #
+    #  changeWeapon - changes the weapons
```

```
+    #                It returns the amount of ammo left.
+    #
+
+
+    def changeWeapon (self, n):
+        return self._cache.changeWeapon (n)
+
+    #
     #  sync - wait for any event to occur.
     #         The event will signify the end of
     #         move, fire, turn, reload action.
diff --git a/python-bot/python_doommarine.py b/python-bot/python_doommarine.py
index 4de6fed..505a4fc 100644
--- a/python-bot/python_doommarine.py
+++ b/python-bot/python_doommarine.py
@@ -30,11 +30,11 @@ def circle ():
     while True:
         for a in range (0, 360, 45):
             runArc (a+180)
-        time.sleep (5)
+        time.sleep (1)
         for w in range (0, 10):
             print "attempting to change to weapon", w,
             print "dhewm3 returns", b.changeWeapon (w)
-            time.sleep (3)
+            time.sleep (1)

 def testturn (a):
     b.turn (a, 1)
@@ -88,7 +88,7 @@ def findAll ():
         if i != me:
             b.aim (i)
             moveTowards (i)
-            time.sleep (5)
+            time.sleep (1)

 def findYou (b):
     for i in b.allobj ():
@@ -104,7 +104,7 @@ def antiClock (b):
         b.turnface (v, 1)
         b.sync ()
         print "waiting"
-        time.sleep (10)
+        time.sleep (2)
         print "next"
         b.reset ()

@@ -117,11 +117,39 @@ def clock (b):
         b.turnface (v, -1)
         b.sync ()
         print "waiting"
-        time.sleep (10)
+        time.sleep (2)
         print "next"
         b.reset ()
```

```
+# Makes bot crouch
+def crouch (b):
+    b.reset ()
+    b.stepup (-5, 3*12)
+    b.select (['move'])
+
+# Makes bot humo
+def jump (b):
+    b.reset ()
+    b.stepup (100, 4*12)
+    b.select (['move'])
+
+# Fires at you
+def fire (i, b):
+    crouch (b)
+    #b.start_firing()
+    #b.stop_firing()
+    #b.reload_weapon()
+
+
+def test_crouch_jump (b):
+    b.reset ()
+    b.stepup (-2, 3*12)
+    b.select (['move'])
+    time.sleep (2)
+    b.stepup (100, 4*12)
+    b.select (['move'])
+
 doommarine = -2

 def execBot (b, useExceptions = True):
@@ -148,8 +176,12 @@ def botMain (b):
     while True:
         moveTowards (you)
         b.face (you)
-        # b.fire ()
-        time.sleep (3)
+        #jump(b)
+        #crouch(b)
+        b.changeWeapon(2)
+        print (b.start_firing())
+        b.stop_firing()
+        time.sleep (0.1)
```

5.4 GIT DIFF: Pybot (C++)

```
diff --git a/neo/game/Player.cpp b/neo/game/Player.cpp
index 8d42dce..cef07f1 100644
--- a/neo/game/Player.cpp
+++ b/neo/game/Player.cpp
@@ -2953,7 +2953,7 @@ void idPlayer::FireWeapon( void ) {
           if ( hud ) {
              hud->HandleNamedEvent( "soulCubeNotReady" );
           }
-          SelectWeapon( previousWeapon, false );
+          //SelectWeapon( previousWeapon, false );
         }
      } else {
        NextBestWeapon();
@@ -6403,9 +6403,11 @@ void idPlayer::Think( void ) {
         buttonMask &= (~ BUTTON_RUN);
         usercmd.rightmove = 0;
         usercmd.forwardmove = 0;
+          usercmd.upmove = 0;
         gameLocal.usercmds[entityNumber].rightmove = 0;
         gameLocal.usercmds[entityNumber].forwardmove = 0;
         gameLocal.usercmds[entityNumber].buttons = 0;
+          gameLocal.usercmds[entityNumber].upmove = 0;
       }
        }
      pulseCount.inc_angle (this);
@@ -6434,7 +6436,7 @@ void idPlayer::Think( void ) {
     }
     usercmd.forwardmove = 0;
     usercmd.rightmove = 0;
-    usercmd.upmove = 0;
+
   }

  // log movement changes for weapon bobbing effects
@@ -6750,9 +6752,9 @@ int idPlayer::Fire (bool firing)
 {
  if (firing)
    {
+    SelectWeapon(1, true);
      buttonMask = 0;
      usercmd.buttons = 0;
-     SelectWeapon (1, true);
      buttonMask |= BUTTON_ATTACK;
      usercmd.buttons |= BUTTON_ATTACK;
    }
@@ -6762,7 +6764,7 @@ int idPlayer::Fire (bool firing)
      usercmd.buttons &= ~(BUTTON_ATTACK);
    }
  if (currentWeapon >= 0 && currentWeapon < MAX_WEAPONS)
-   return inventory.ammo[currentWeapon];
+   return inventory.ammo[currentWeapon + 1];
  return 0;
 }
```

```
@@ -6778,18 +6780,37 @@ int idPlayer::ChangeWeapon (int new_weapon)
   inventory.weapons = -1;
   if (new_weapon >= 0 && new_weapon < MAX_WEAPONS)
     {
-      if ((inventory.weapons & (1 << new_weapon)) != 0)
- {
-   /*
-    * player is carrying this weapon.
-    */
-   SelectWeapon (new_weapon, true);
-   return inventory.ammo[currentWeapon];
- }
+       SelectWeapon (new_weapon, true);
+       return inventory.ammo[currentWeapon];
+        if ((inventory.weapons & (1 << new_weapon)) != 0)
+      {
+        /*
+         * player is carrying this weapon.
+         */
+      }
     }
   return -1;
 }

+/*
+================
+idPlayer::reload_weapon
+================
+*/
+int idPlayer::reload_weapon (void) {
+  if ( gameLocal.isClient ) {
+    return -1;
+  }
+  if ( spectating || gameLocal.inCinematic || influenceActive ) {
+    return -1;
+  }
+  if ( weapon.GetEntity() && weapon.GetEntity()->IsLinked() ) {
+    weapon.GetEntity()->Reload ();
+    return inventory.ammo[currentWeapon];
+  }
+  return -1;
+}
+

 /*
 ==============
@@ -6806,6 +6827,29 @@ int idPlayer::Ammo (void)


 /*
+==============
+idPlayer::stepUp (crouch or jump)
+==============
+ */
+int idPlayer::stepUp (int vel, int dist)
+{
```

```
+  int old = (int) usercmd.upmove;
+
+  usercmd.upmove = (signed char) vel;
+  usercmd.forwardmove = 0;
+  usercmd.rightmove = 0;
+  buttonMask = 0;
+  buttonMask |= BUTTON_RUN;
+  pulseCount.set_run (dist);
+  usercmd.buttons |= BUTTON_RUN;
+  gameLocal.usercmds[entityNumber] = usercmd;
+
+  return old;
+}
+
+
+/*
 ==============
 idPlayer::Kill
 ==============
diff --git a/neo/game/ai/pybot.cpp b/neo/game/ai/pybot.cpp
index 12914f4..9735a64 100644
--- a/neo/game/ai/pybot.cpp
+++ b/neo/game/ai/pybot.cpp
@@ -124,13 +124,14 @@ class item
   int stepForward (int vel, int dist);
   int stepRight (int vel, int dist);
   int stepVec (int velforward, int velright, int dist);
+  int stepUp (int vel, int dist);
   int start_firing (void);
   int stop_firing (void);
   int ammo (void);
   int weapon (int new_weapon);
   int health (void);
   int angle (void);
-  void reload_weapon (void);
+  int reload_weapon (void); // Khalid
   bool aim (idEntity *enemy);
   int turn (int angle, int angle_vel);
   idEntity *getIdEntity (void);
@@ -274,6 +275,24 @@ int item::stepForward (int vel, int dist)
   return 0;
 }

+/*
+ *  stepUp - step up at velocity, vel, and over distance, dist.
+ */
+
+int item::stepUp (int vel, int dist)
+{
+  switch (kind)
+    {
+#if 0
+    case item_monster:
+      return idai->StepDirection (vel, dist);
+#endif
+    case item_player:
```

```
+     return idplayer->stepUp (vel, dist);
+   }
+  return 0;
+}
+

 /*
  *  stepForward - step forward at velocity, vel, and over distance, dist.
@@ -398,6 +417,16 @@ int item::stop_firing (void)

 int item::ammo (void)
 {
+  switch (kind)
+    {
+    case item_monster:
+      assert (false);
+      return 0; // ignore
+      break;
+    case item_player:
+      return idplayer->Ammo();
+    }
+  assert (false);
   return 0;
 }

@@ -446,9 +475,19 @@ int item::angle (void)
  *  reload_weapon
  */

-void item::reload_weapon (void)
+int item::reload_weapon (void)
 {
-
+  switch (kind)
+    {
+    case item_monster:
+      assert (false);
+      return 0; // ignore
+      break;
+    case item_player:
+      return idplayer->reload_weapon ();
+    }
+  assert (false);
+  return 0;
 }


@@ -500,6 +539,7 @@ class dict
   bool stepDirection (int id, float dir);
   int stepForward (int id, int vel, int units);
   int stepRight (int id, int vel, int dist);
+  int stepUp (int id, int vel, int dist);
   int stepVec (int id, int velforward, int velright, int dist);
   int start_firing (int id);
   int stop_firing (int id);
@@ -634,6 +674,15 @@ int dict::stepForward (int id, int vel, int dist)
```

```
   return entry[id]->stepForward (vel, dist);
 }

+/*
+ *  setUp - step up by, units.
+ */
+
+int dict::stepUp (int id, int vel, int dist)
+{
+  return entry[id]->stepUp (vel, dist);
+}
+

 /*
  *  stepVec - step forward and right simulataneously by dist units.
@@ -664,6 +713,17 @@ int dict::stop_firing (int id)
   return entry[id]->stop_firing ();
 }

+/*
+ *  reload_weapon - reload the current weapon and return the
+ *                  ammo available for the current weapon.
+ */
+
+
+int dict::reload_weapon (int id)
+{
+  return entry[id]->reload_weapon ();
+}
+

 /*
  *  ammo - return the ammo available for the current weapon.
@@ -1647,6 +1707,8 @@ void pyBotClass::interpretRemoteProcedureCall (char *data)
     rpcForward (&data[8]);
   else if (idStr::Cmpn (data, "stepvec ", 8) == 0)
     rpcStepVec (&data[8]);
+  else if (idStr::Cmpn (data, "step_up ", 8) == 0)
+    rpcStepUp (&data[8]);
   else if (strcmp (data, "start_firing") == 0)
     rpcStartFiring ();
   else if (strcmp (data, "stop_firing") == 0)
@@ -1865,6 +1927,34 @@ void pyBotClass::rpcForward (char *data)
   state = toWrite;
 }

+/*
+ *  rpcStepUp - step up.
+ *              The parameter, data, contains two parameters: velocity and distance.
+ */
+
+void pyBotClass::rpcStepUp (char *data)
+{
+  char buf[1024];
+  int vel = 0;
+  int dist = 0;
```

```
+
+  if (protocol_debugging)
+    gameLocal.Printf ("rpcStepUp (%s) call by python\n", data);
+
+  if (rpcId > 0)
+    {
+      vel = atoi (data);
+      char *p = index (data, ' ');
+      if ((p == NULL) || ((*p) == '\0'))
+  dist = 0;
+      else
+  dist = atoi (p);
+      dist = dictionary->stepUp (rpcId, vel, dist);
+    }
+  idStr::snPrintf (buf, sizeof (buf), "%d\n", dist);
+  buffer.pyput (buf);
+  state = toWrite;
+}

 /*
  *  rpcStepVec - step forward along a vector.
@@ -2109,7 +2199,7 @@ void pyBotClass::rpcReloadWeapon (void)
     gameLocal.Printf ("rpcReloadWeapon call by python\n");

   if (rpcId > 0)
-    ammo = dictionary->ammo (rpcId); // --fixme-- this should call something else
+    ammo = dictionary->reload_weapon (rpcId);
   else
     ammo = 0;
```