

Implementing Jump and Crouch in the Python API

- the files which need to be modified are split into two groups
 - server side (dhewm3 engine)
 - client side (python)
- server side files: `neo/game/Player.cpp`,
`neo/game/Player.h`,
`neo/game/ai/pybot.cpp`
`neo/game/ai/pybot.h`
- client side files: `python-bot/botbasic.py`,
`python-bot/botcache.py`, `python-`
`bot/botlib.py`, `python-`
`bot/python_doommarine.py`

Test code: `python_doommarine.py`

- add this code somewhere appropriate in `python_doommarine.py` and ensure that it is called!
- `python-bot/python_doommarine.py`

```
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'])
```

Implementing Jump and Crouch in the Python API

- these are guidelines on how to implement `stepUp` (crouch and jump)
- the notes will require you to add extra code on the server side, the client should be complete

Test code: `python_doommarine.py`

- can see the new method `stepvec` which is a Z-axis movement
 - in doom3 the Z-axis movement, results in the player self returning back to the neutral position
 - gravity pulls the player back to the ground, and likewise happens if the player crouches

stepup in botlib.py

- add stepup after the method right

python-bot/botlib.py

```
#
# stepup - makes the bot jump or crouch.
#
def stepup (self, velup, dist):
    return self._cache.stepup (velup, dist)
```

stepup in botlib.py

python-bot/botcache.py

```
#
# stepup -
#
def stepup (self, velup, dist):
    self.delpos (self.me ())
    return self._basic.stepup (velup, dist)
```

python-bot/botbasic.py

python-bot/botbasic.py

```
#
# 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)
```

Server side changes

- add the declaration of the method stepUp

neo/game/Player.h

```
bool isVisible (idEntity *enemy);
int Fire (bool firing);
int Ammo (void);
int stepUp (int vel, int dist);
int Turn (int angle, int angle_vel);
int GetYaw (void);
void doTurn (int angle);
void select (int bitmask);
int ChangeWeapon (int new_weapon);
```

stepUp implementation inside Player.cpp

neo/game/Player.cpp

```

/*
=====
idPlayer::stepUp (crouch or jump)
=====
*/
int idPlayer::stepUp (int vel, int dist)
{
    int old = (int) usercmd.upmove;

    // gameLocal.Printf( "stepUp %d %d0, vel, dist );
    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;
}

```

stepUp implementation inside Player.cpp

- note that in order for the stepUp to function correctly it will require changes made to the Think method in the same file

- you should assign

```

usercmd.upmove = 0;
gameLocal.usercmds[entityNumber].upmove = 0;

```

- when get_run () == 0

stepUp implementation inside Player.cpp

- now you need to make changes to pybot.cpp to interpret the step_up string passed by python and decode its two parameters
 - follow though the reload_weapon changes to see which classes need to be changed
- it might be useful to examine the stepVec method and how it is implemented
 - your stepUp method should be similar