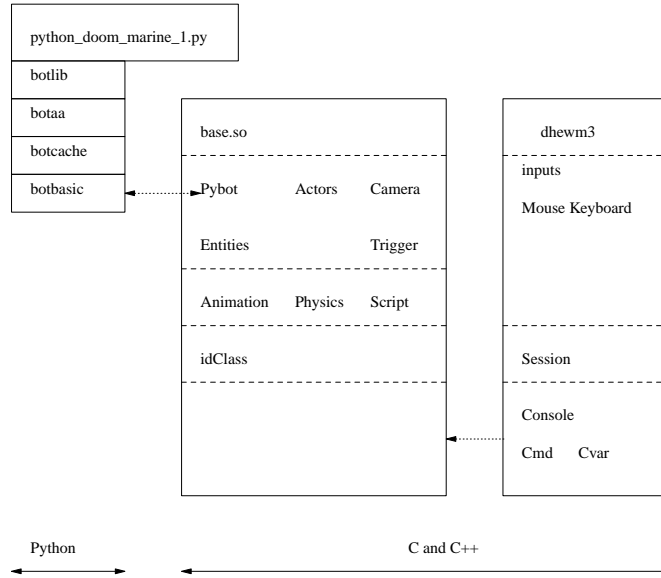


# Implementing Select Weapon in the Python API



# Implementing Select Weapon in the Python API

■ \$HOME/Sandpit/git-doom3/pybot-dhewm3/python-bot/botbasic.p

```
#
# changeWeapon - change to weapon, n.
#                 Attempt to change to weapon, n.
#                 n is a number 0..maxweapon
#                 The return value is the amount
#                 of ammo left for the weapon
#                 >= 0 if the weapon exists
#                 or -1 if the weapon is not in
#                 the bots inventory.
#
```

# Implementing Select Weapon in the Python API

■ \$HOME/Sandpit/git-doom3/pybot-dhewm3/python-bot/botbasic.p

```
def changeWeapon (self, n):
    if debug_protocol:
        print "requesting change weapon to", n
    s = "change_weapon %d\n" % (n)
    self.s.send (s)
    l = self.getLine ()
    if debug_protocol:
        print "doom returned", l
    return int (l)
```

## Test code for the python bot

- `$HOME/Sandpit/git-doom3/pybot-dhewm3/python-bot/python_doo`

```
b = botbasic.basic ("localhost", "python_doommarine_1")
print "success! python doom marine is alive"
while True:
    for w in range (1, 8):
        print "attempting to change to weapon", w,
        print "dhewm3 returns", b.changeWeapon (w)
        time.sleep (3)
```

## Pybot.cpp server side change

- `$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp`

```
else if (idStr::Cmpn (data, "get_pair_name_entity ", 21) == 0)
    rpcGetPairEntity (&data[21]);
else if (idStr::Cmpn (data, "get_entity_pos ", 15) == 0)
    rpcGetEntityPos (&data[15]);
else if (idStr::Cmpn (data, "change_weapon ", 14) == 0)
    rpcChangeWeapon (&data[14]);
else
{
    gameLocal.Printf ("data = \"%s\"", len (data) = %d\n
    ERROR ("unrecognised rpc command");
}
```

- notice how we check for the appropriate method to call `rpcChangeWeapon` by examining the first 14 characters of data

## Method declaration in pybot.h

- `$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.h`

```
void rpcGetPairEntity (char *arg);
void rpcGetEntityPos (char *data);
void rpcChangeWeapon (char *data);
int myid;
char *name;
```

## Implementation in pybot.cpp

- `$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp`

```
/*
 * rpcChangeWeapon - attempt to change weapon to the number
 *                   The amount of ammo is returned. -1
 */

void pyBotClass::rpcChangeWeapon (char *data)
{
    if (protocol_debugging)
        gameLocal.Printf ("rpcChangeWeapon (%s) call by python");

    char buf[1024];
    int weapon = atoi (data);
    int ammo = -1;

    if (weapon >= 0)
        ammo = dictionary->weapon (myid, weapon);
    idStr::snPrintf (buf, sizeof (buf), "%d\n", ammo);
    if (protocol_debugging)
        gameLocal.Printf ("rpcChangeWeapon responding with: %s", buf);
    buffer.pyput (buf);
    state = toWrite;
}
```

**dict::weapon**

- \$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp

```
bool aim (int id, int enemy);
int turn (int id, int angle, int angle_vel);
void select (int id, int mask);
int getHigh (void);
int weapon (int id, int new_weapon);
private:
    item *entry[MAX_ENTRY];
    int high;
```

**dict::weapon**

- \$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp

```
int dict::turn (int id, int angle, int angle_vel)
{
    return entry[id]->turn (angle, angle_vel);
}

/*
 * weapon - change to new_weapon and return the amount of
 * ammo for this weapon. -1 if the weapon is not in the
 * inventory.
 */

int dict::weapon (int id, int new_weapon)
{
    return entry[id]->weapon (new_weapon);
}
```

**item::weapon declaration**

- \$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp

```
int stepVec (int velforward, int velright, int dist);
int start_firing (void);
int stop_firing (void);
int ammo (void);
int weapon (int new_weapon);
int health (void);
void reload_weapon (void);
bool aim (idEntity *enemy);
```

**item::weapon implementation**

- \$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/ai/pybot.cpp

```
/*
 * weapon - attempt to select weapon, new_weapon.
 * If successful return the amount of ammo else
 */

int item::weapon (int new_weapon)
{
    switch (kind)
    {
        case item_player:
            return idplayer->ChangeWeapon (new_weapon);
    }
    assert (false);
    return 0;
}
```

## idPlayer::weapon implementation

- \$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/Player.cpp

```

/*
=====
idPlayer::ChangeWeapon (gaius) (see StealWeapon)
=====
*/

int idPlayer::ChangeWeapon (int new_weapon)
{
    inventory.weapons = -1; // testing only!
    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];
        }
    }
    return -1;
}

```

## idPlayer::weapon prototype

- \$HOME/Sandpit/git-doom3/pybot-dhewm3/neo/game/Player.h

```

int Ammo (void); // gaius
int Turn (int angle, int angle_vel); // gaius
void doTurn (int angle); // gaius
void select (int bitmask); // gaius
int ChangeWeapon (int new_weapon);

private:

```

## Conclusion

- comment this code and see if you can extend the API to include other features