

## Finishing implementation of health (server side)

- so far the Python bot can call the server and we always get the value 42!
- need to find the real value of the server side bot health
- the health is contained inside the structure `gentity_t` and can be accessed via the `g_entities` array

slide 3  
gaius

## Finishing implementation of health (server side)

- `$HOME/Sandpit/ioquake-latest/ioquake3/code/game/ai_main.c`

```
void
call_trap_rpc (int client, int character)
{
    g_entities[client].info.health = g_entities[client].health;
    trap_CheckRPC(client, character, &g_entities[client].info.health);
}
```

- notice that `call_trap_rpc` copies the health value into an parameter area before calling `trap_CheckRPC`

slide 4  
gaius

## New file

- `$HOME/Sandpit/ioquake-latest/ioquake3/code/botlib/pybotinfo.h`

```
#if !defined (PYBOTINFO_H)
# define PYBOTINFO_H

typedef struct pybotinfo_s {
    int health;
    // you can obviously extend this struct to contain more
} pybotinfo;

#endif
```

**New file**

- this is included from within `g_local.h`
- `$HOME/Sandpit/ioquake-latest/ioquake3/code/game/g_local.h`

```
#if !defined(G_LOCAL_H)
# define G_LOCAL_H
#include "../qcommon/q_shared.h"
#include "bg_public.h"
#include "g_public.h"
#include "../botlib/pybotinfo.h"
...
```

**be\_ai\_char.c**

- `$HOME/Sandpit/ioquake-latest/ioquake3/code/botlib/be_ai_ch`

```
int dohealth (void *p)
{
    py_bot_t *py = (py_bot_t *)p;

    returnInt (p, py->info->health);
    return qtrue;
}
```

**Obtaining a copy with health implemented server side**

- you dont need to get this copy, but if you want the all health changes then this version has the complete changes applied

- ```
$ ssh mcgreg.comp.glam.ac.uk
<enter your linux password>
$ cd $HOME/Sandpit
$ wget http://floppsie.comp.glam.ac.uk/download/c/ioquake
$ rm -rf ioquake-latest
$ tar zxf ioquake-20161114.tar.gz
$ exit
# your command line is back on the client
$ cd $HOME/Sandpit/ioquake-latest3/ioquake
$ ./compilequake
```