

## Extending the Remote procedure call API

- `disableAI () : boolean`
  - disable the in game C AI
  - return True/False if successful
  - (completed)

## Extending the Remote procedure call API

- `enableAI () : boolean`
  - enable the in game C AI
  - return True/False if successful
  - (completed)

## Implementing these two function calls

- we need to modify the Python remote procedure call interface
- then we need to modify the ioquake source code
  - (completed)

## Disable/Enable ioquake AI

- this has been completed in your source file
  - these notes show how the changes were made and
  - importantly also show you which files were altered

## Python

- open up ioquake-latest/python-bot/bot-legoman/botfiles/bots/botlib.py
- alter

```
# AI codes
SKILL, CONT = range(1, 3)
```

## Python

- to

```
# AI codes
SKILL, CONT, CAI = range(1, 4)
```

- we now are going to define this new function CAI

## Python

- now move to the definition of class bot: and continue down to find def cont (self):
- we now add

```
def cai (self, boolean):
    """ enables/disables the C AI engine inside ioquake """
    """ It returns the previous value of the C AI engine """
    """ True means it was on, False means it was off """
    return calliB(AI, CAI, boolean)
```

# Python

- calliB passes the three parameters as integers and returns a boolean result

- we now implement two more Python functions:

```
def disableAI ():
    """ disable the C AI and return True if successful """
    return cai(False)=True

def enableAI ():
    """ enable the C AI and return True if successful """
    return cai(True)=False
```

- the Python code is complete!

# C

- open up the file: ioquake-  
latest/ioquake3/code/botlib/be\_ai\_char.c

```
/* a bot character
typedef struct bot_character_s
{
    char filename[MAX_QPATH];
    float skill;
    int isPythonBot;
    py_bot_t py;
    bot_characteristic_t c[1];
} bot_character_t;
```

# C

- and change it to:

```
/* a bot character
typedef struct bot_character_s
{
    char filename[MAX_QPATH];
    float skill;
    int isPythonBot;
    int c_ai;
    py_bot_t py;
    bot_characteristic_t c[1];
} bot_character_t;
```

# C

- change:

```
#if 1
if ((strlen(charfile)>3) && (strcmp(&charfile[strlen(charfile)-3], "py") != 0)) {
    ch->isPythonBot = qtrue;
    if (initPy(&ch->py, ch)) {
        ch->skill = skill;
    }
}
```



- to

```

■ 
#if 1
ch->c_ai = qtrue;
if ((strlen(charfile)>3) && (strcmp(&charfile[strlen(charfile)-3], "py") == 0)) {
ch->isPythonBot = qtrue;
if (initPy(&ch->py, ch)) {
ch->skill = skill;
}
}
}


```



- add the new function underneath `contrpc`



```

■ 
/*
 * cairpc - called by the rpc.
 *          Bytes: <length><CODE><FUNCTION>
 *          Integer: <True/False>
 *          The Integer value turns the C AI on/off.
 */

int cairpc (void *p)
{
    py_bot_t *py = (py_bot_t *)p;
    int *onoff = (int *)&py->inBuf[3];
    bot_character_t *ch = py->ch;
    int oldValue = ch->c_ai;

    ch->c_ai = *onoff;
    returnBoolean(p, (unsigned char)oldValue);
    return qtrue;
}


```



- open up `ioquake3/code/botlib/be_ai_char.h` and add this prototype to the end of the file:

```

■ 
/*
 * use_c_ai - returns true if, client, should use the C AI.
 */
int use_c_ai (int character);


```



- now open up ioquake-  
latest/ioquake3/code/botlib/be\_ai\_char.c  
and add this function after isPythonBot

```

/*
 * use_c_ai - returns true if, client, should use the C :
 */

int use_c_ai (int character)
{
    return (botcharacters[character] != NULL) && (botcharac
}

```



- now we need to add the C AI rpc call, so open up  
ioquake-  
latest/ioquake3/code/botlib/be\_ai\_py.h  
and alter

```

/*
 * AI codes
 */

typedef enum Aicode_t {
    pyAIDummy, pySKILL, pyCONT, pyAIImax,
} Aicode;

```



- to

```

/*
 * AI codes
 */

typedef enum Aicode_t {
    pyAIDummy, pySKILL, pyCONT, pyCAI, pyAIImax,
} Aicode;

```



- alter the prototype for function initAImethods to

```

/*
 * initAImethods - initialise the AI methods called by t
 */

EXTERN void initAImethods (py_bot_t *py,
                           int (*skill)(void *),
                           int (*cont)(void *),
                           int (*cai)(void *));

```



## Exercise for the reader

- now we need to modify the call to `initAImethods` (found in file `be_ai_char.c`)

- ```
if (initPy(&ch->py, ch)) {
    ch->skill = skill;
    initAImethods(&ch->py, skillrpc, contrpc, cairpc);
}
```

- at this point the Python rpc mechanism can turn off/on the `c_ai` value
- the C `ioquake` code can use the function `use_c_ai` to test this bit
- we still need to modify `ioquake` so that it does not call the C AI if this function returns `TRUE`