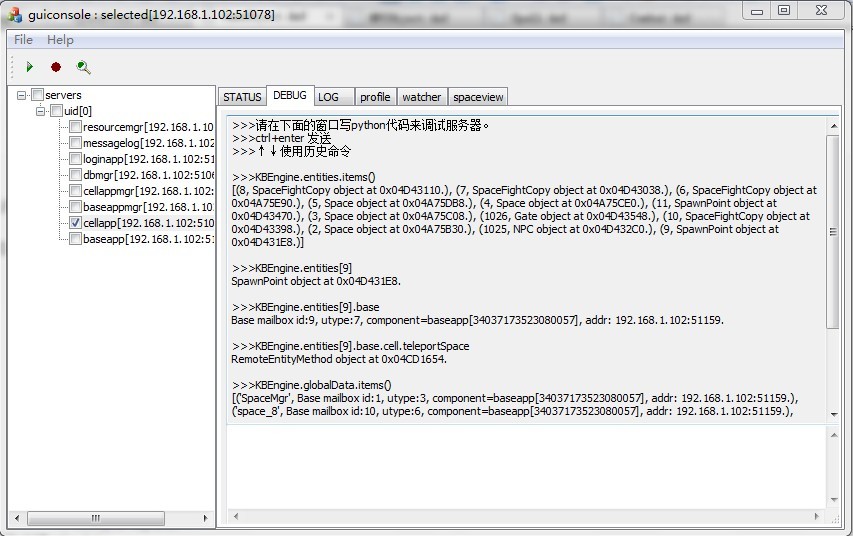
http://kbengine.org/cn/docs/documentations/onlinedebugging.html

在线调试工具:

使用可视化工具[GUIConsole](http://www.kbengine.org/cn/docs/tools/guiconsole.html):



使用脚本命令工具[Cluster Controller](http://www.kbengine.org/cn/docs/tools/pycluster.html):

* 你可以使用这个工具查看服务端运行的一些状态信息 (参看: [Information query servers](http://www.kbengine.org/cn/docs/tools/pycluster.html))
* 你可以使用这个工具进入Python命令行调试 (参看: [Console](http://www.kbengine.org/cn/docs/tools/pycluster.html))

使用telnet服务[kbengine\_defs.xml](http://www.kbengine.org/cn/docs/configuration/kbengine_defs.html)->telnet\_service:

cellapp : telnet localhost 50000

baseapp : telnet localhost 40000

client : telnet localhost 51000

调试技巧:

###允许系统产生core文件(仅Linux):

在~/.bashrc中添加如下命令:

ulimit -c unlimited

需要root权限执行:

[root@gameserver ~]# echo '%e.core.%p' > /proc/sys/kernel/core\_pattern

###断点调试:

仅引擎层c++代码可以使用断点调试，断点调试请先关闭服务端心跳机制[kbengine\_defs.xml](http://www.kbengine.org/cn/docs/configuration/kbengine_defs.html)->channelCommon->timeout.

脚本层只能查看输出日志，或者使用Python命令行来调试，由于是分布式服务程序没有增加断点的功能。

###Python命令行调试游戏逻辑例子(在Python命令行输入):

查看当前进程上的所有Entity:

>>> KBEngine.entities.items()

[1: Space at 0x4D3040, 2: Monster at 0x4D3038]

>>> for entityID, entity in KBEngine.entities.items():

print("entityID:%i, entity=%s", entityID, entity)

1, Space at 0x4D3040

2, Monster at 0x4D3038

查看Entity当前的坐标:

>>> KBEngine.entities[entityID].position

(10.0, 0, 10.0)

改变Entity的朝向:

>>> KBEngine.entities[entityID].direction.z = math.pi

调用Entity的接口:

>>> KBEngine.entities[entityID].funcXXX()

手动创建一个Entity(cellapp):

>>> e = KBEngine.createEntity("Monster", spaceID, (10.0, 0, 10.0), (0.0, 0, 0.0), {})

调用一个Entity的远程方法(cellapp):

>>> KBEngine.entities[entityID].base.func()

>>> KBEngine.entities[entityID].client.func()

查看全局变量

KBEngine.globalData.items()

查看内存泄漏

KBEngine.entities.garbages.items()

你可以在Python命令行输入任意的Python语句并执行。

###性能分析:

引擎性能分析

* [GUIConsole](http://www.kbengine.org/cn/docs/tools/guiconsole.html)->profile选项卡 : 选择cprofile进行分析。
* [Cluster Controller](http://www.kbengine.org/cn/docs/tools/pycluster.html) : 使用”:cprofile”命令。

脚本性能分析

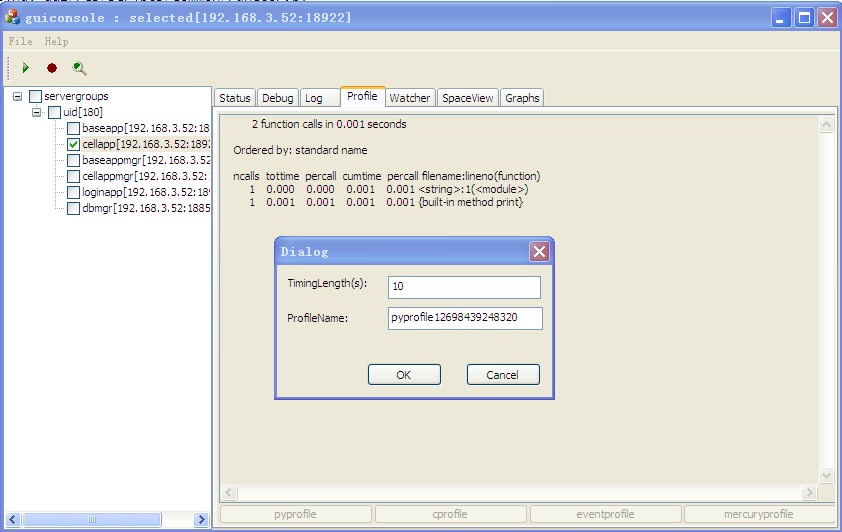
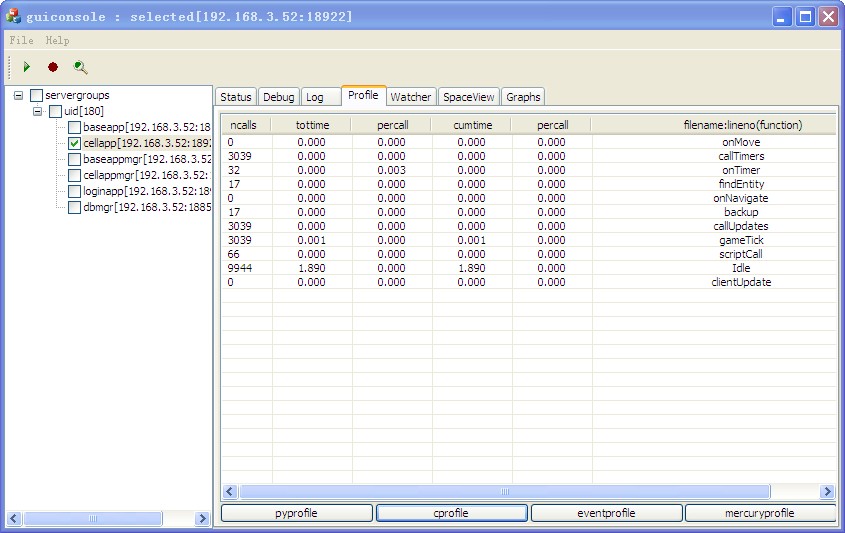
* [GUIConsole](http://www.kbengine.org/cn/docs/tools/guiconsole.html)->profile选项卡 : 选择pyprofile进行分析。
* [Cluster Controller](http://www.kbengine.org/cn/docs/tools/pycluster.html) : 使用”:pyprofile”命令。

网络状态分析

* [GUIConsole](http://www.kbengine.org/cn/docs/tools/guiconsole.html)->profile选项卡 : 选择networkprofile进行分析。
* [Cluster Controller](http://www.kbengine.org/cn/docs/tools/pycluster.html) : 使用”:networkprofile”命令。

事件处理分析

* [GUIConsole](http://www.kbengine.org/cn/docs/tools/guiconsole.html)->profile选项卡 : 选择eventprofile进行分析。
* [Cluster Controller](http://www.kbengine.org/cn/docs/tools/pycluster.html) : 使用”:eventprofile”命令。

###监视变量:

引擎允许使用工具监视当前进程上默认提供的变量(例如:总发包数，当前在线的玩家数)，同时用户也可以在脚本中添加需要监视的变量。

查看方式: [GUIConsole](http://www.kbengine.org/cn/docs/tools/guiconsole.html)->watcher选项卡。

添加脚本监视变量的例子(监视在线玩家数):

baseapp->watcher.py:

...

...

def countPlayers():

i = 0

for e in KBEngine.entities.values():

if e.\_\_class\_\_.\_\_name\_\_ == "Avatar":

i += 1

return i

...

...

KBEngine.addWatcher("scripts/countPlayers", "UINT32", countPlayers)

...

...

