Hearthbuddy\Triton\Game\TritonHs.cs

游戏主界面的4/5个按钮点击函数

Hearthbuddy\Triton\Bot\Logic\Bots\DefaultBot\DefaultBot.cs

private async Task method\_45(PegasusScene **pegasusScene\_0**)

[HubScene]按收藏按钮等过程

Hearthstone.exe -launch -uid hs\_beta

Battle.net.exe --exec="launch WTCG"

Ls

更改文件：ass ：

Logger print():

加入了写入c//1//1.txt

注释了两个地方的限制输出

LoginService update():

判断请求登录状态，登录

LoginService ProcessChallenge(url):

有一个url，可能是请求的地址

可能是请求，获取token吧

DesktopLoginTokenFetcher FetchToken(url)

获取token等可能是设置本地token

GetTokenIfAvailable

获取token，一种是GetTokenFromTokenFetcher

一种是BattleNet.GetLaunchOption

第一种已弃用

registrywin

RetrieveString()

计算机\HKEY\_CURRENT\_USER\Software\Blizzard Entertainment\Battle.net\Launch Options\WTCG

using System.IO;

using System.Text;

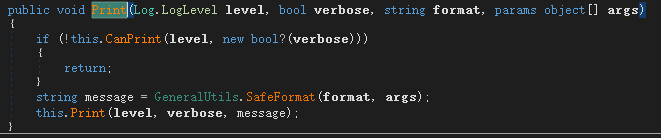
                StreamWriter streamWriter = new StreamWriter("C:\\1\\11.txt", true, Encoding.Default);  
                streamWriter.WriteLine(**path**+"  "+**name**);  
                streamWriter.Flush();  
                streamWriter.Close();

TavernBrawlDisplay OnPlayButtonExecute()

如何让ls输出所有调试信息呢？

先反编译Assembly-CSharp.dll(位置：Hearthstone目录\Hearthstone\_Data\Managed)（工具dnspy）

找到Logger类中的print函数发现输出日志时

​

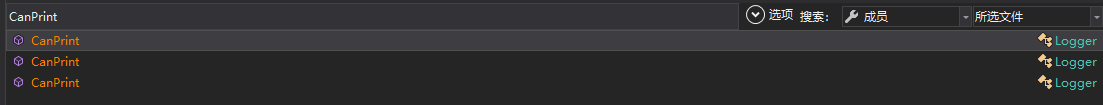
有CanPrint限制，说明在exe中指定了某些配置通过CanPrint的判断让很多log没有输出

右键-编辑方法-删除if语句（注意备份）

发现炉石log下的日志没变（。。。。。？）

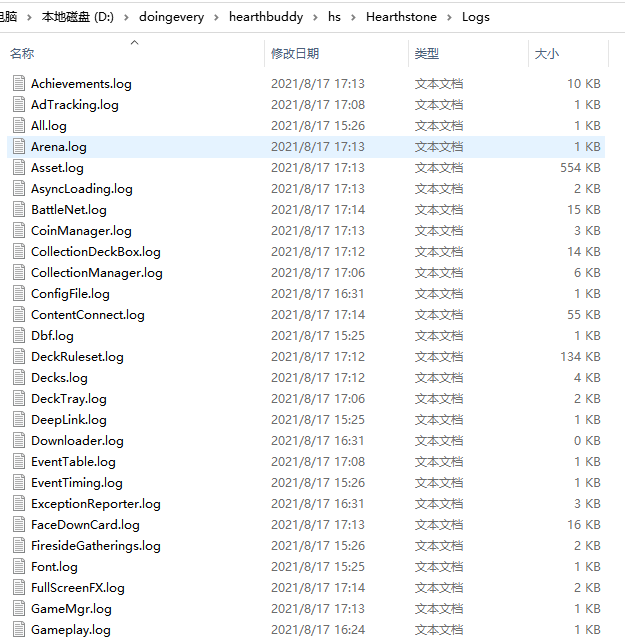
遂怀疑在日志print链条内还进行了CanPrint判断

dnspy搜索CanPrint，（注意，搜索时不要选择所有文件，因dnspy使用的是递归搜索，不指定范围可能把整个net框架给搜索一便（这也是近两个版本dnspy容易崩溃的原因））



将每一个CanPrint第一行直接返回True，保存，ok！

结果：



这样我们就能很方便的对ls进行定位了测试了

声明：此行为仅可用于学习使用，请各位健康游戏

再给大家讲一下游戏启动的问题吧，因为中控是最好写的，可以用任何一个语言实现，相信大家都尝试过

但大家肯定发现现在跟以前不太一样了，不能从ls（炉石）的exe直接启动

为什么呢？

​通过日志的报错信息，我们补全参数

Hearthstone.exe -launch -uid hs\_beta

发现还是不行

怎么回事呢？

先反编译Assembly-CSharp.dll（dnspy）

通过log追踪函数发现ls是用token验证登录的，定位到token获取函数后发现有两种方式：

1通过请求方式

2获取注册表内的token值

发现方式1一直没有使用，猜测方法1已经弃用

追踪注册表指定值

发现在暴雪界面每按一下进入游戏就会更新一次注册表的值，然后ls就会读取注册表内的信息，然后获取token，提交请求实现登录。

结论：无法从ls的exe文件直接启动

解决方法：

带参数运行战网exe：

Battle.net.exe --exec="launch WTCG"

hb：

点击运行按钮到运行脚本逻辑的框架：

MainWindow.button\_0\_Click():

BotManager.Start();//开始bot脚本

BotManager.Start():

BotThread = new Thread(smethod\_4);

BotThread.Name = "BotThread";

BotThread.Start();

BotManager.smethod\_4():

1.

smethod\_0(CurrentBot);

->ibot\_1.Start();

->DefaultBots.Start():

->coroutine\_0 = new Coroutine(() => method\_21())

定义了taskProducer异步委托:method\_21()

定义了action\_0异步委托：里面有taskProducer()

2.

smethod\_1(CurrentBot);

这里实现的循环：

while (!autoResetEvent\_0.WaitOne(0))

{

try

{

if (MsBeforeNextTick != 0)

{

Thread.Sleep(MsBeforeNextTick);

MsBeforeNextTick = 0;

}

smethod\_1(CurrentBot);//这里

bool\_3 = false;

if (MsBetweenTicks != 0)

{

Thread.Sleep(MsBetweenTicks);

}

}

catch (InjectionDesyncException)

{

ilog\_0.DebugFormat("[BotThreadFunction] An InjectionDesyncException was detected.");

bool\_3 = true;

TritonHs.InvokeEvent(eventHandler\_1, null, new ClientFrozenEventArgs());

}

catch

{

}

}

BotManager..smethod\_1(IBot ibot\_1):

ibot\_1.Tick();//DefaultBots.Tick

DefaultBots.Tick():

coroutine\_0.Resume();//bot动作

coroutine\_0.Resume()

coroutine\_0.method\_0(bool\_2: false);

private void method\_0(bool bool\_2)

{

SynchronizationContext current = SynchronizationContext.Current;

Coroutine coroutine = coroutine\_0;

try

{

coroutine\_0 = this;

SynchronizationContext.SetSynchronizationContext(null);

Action action = action\_0;

action\_0 = null;

action();

if (!bool\_2)

{

Ticks++;

}

method\_2(bool\_2);

}

finally

{

coroutine\_0 = coroutine;

SynchronizationContext.SetSynchronizationContext(current);

}

}

action会进入 action\_0异步委托：里面有taskProducer()

taskProducer会进入DefaultBots.method\_21()

后面会进入多个async

最后会

private static async Task smethod\_3()

{

Class65 @class = Current.class65\_0.GetAwaiter();

if (!@class.Boolean\_1)

{

await @class;

//object obj = default(object);

//@class = (Class65)obj;

}

@class.method\_1();

}

进入自定义的awaiter

自定义的awaiter无任何任务，只是将await后面需要运行的通过await内的异步方法生成器掉用awaiter的OnCompleted

public void OnCompleted(Action continuation)

{

if (Current.action\_0 != null)

{

Current.bool\_0 = true;

}

Current.action\_0 = continuation;

}

重新定义Current.action\_0

在下一个循环会运行之前await的后半部分

然后进入DefaultBots.method\_21()

private async Task method\_21()

{

while (true)

{

await Coroutine.Yield();

PegasusScene scene = SceneMgr.Get().GetScene();

SceneMgr.Mode mode = SceneMgr.Get().GetMode();

if (!(await method\_20()))

{

switch (mode)

{

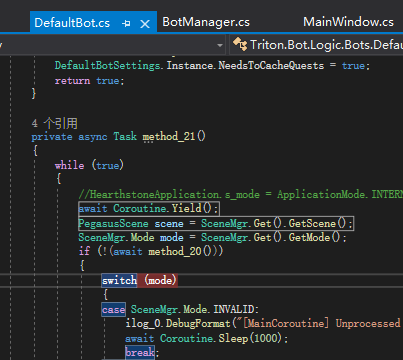
case SceneMgr.Mode.INVALID:

ilog\_0.DebugFormat("[MainCoroutine] Unprocessed mode: {0}.", mode);

await Coroutine.Sleep(1000);

break;

到达await后循环之前的过程



intptr\_1 = intptr\_0 + 0x31060;//94 5D 00030CA0 mono\_class\_get

intptr\_2 = intptr\_0 + 0x32080;//111 6E 00031CC0 mono\_class\_get\_name

intptr\_3 = intptr\_0 + 0x32090;//112 6F 00031CD0 mono\_class\_get\_namespace

intptr\_4 = intptr\_0 + 0x28D70;//21 14 000289A0 mono\_assembly\_get\_image

intptr\_5 = intptr\_0 + 0x320A0;//113 70 00031CE0 mono\_class\_get\_nested\_types

intptr\_6 = intptr\_0 + 0x2B200;//47 2E 0002ACE0 mono\_assembly\_open

intptr\_7 = intptr\_0 + 0x29550;//30 1D 00029180 mono\_assembly\_load\_from\_full

intptr\_8 = intptr\_0 + 0x318E0;//103 66 00031520 mono\_class\_get\_fields

intptr\_9 = intptr\_0 + 0x26090;//274 111 00025CC0 mono\_domain\_get

intptr\_10 = intptr\_0 + 0xEF800;//371 172 000EE030 mono\_gchandle\_free

intptr\_11 = intptr\_0 + 0x9C230;//320 13F 0009B570 mono\_field\_get\_value\_object

intptr\_12 = intptr\_0 + 0xEFB90;//374 175 000EE3C0 mono\_gchandle\_new

intptr\_13 = intptr\_0 + 0x5D350;//481 1E0 0005CC10 mono\_image\_open\_from\_data\_with\_name

intptr\_14 = intptr\_0 + 0x5C4D0;//466 1D1 0005BD90 mono\_image\_get\_table\_rows

intptr\_15 = intptr\_0 + 0x63120;//665 298 000629F0 mono\_method\_signature

intptr\_16 = intptr\_0 + 0x32470;//120 77 000320A0 mono\_class\_get\_type

intptr\_17 = intptr\_0 + 0x80937;//intptr\_17 = intptr\_0 + 0x80937;

intptr\_18 = intptr\_0 + 0xB3C90;//946 3B1 000B27F0 mono\_thread\_detach

intptr\_19 = intptr\_0 + 0x32010;//110 6D 00031C50 mono\_class\_get\_methods

intptr\_20 = intptr\_0 + 0x320F0;//114 71 00031D30 mono\_class\_get\_nesting\_type

intptr\_21 = intptr\_0 + 0x392E0;//301 12C 00038DD0 mono\_event\_get\_raise\_method

intptr\_22 = intptr\_0 + 0x9EE40;//710 2C5 0009E130 mono\_object\_unbox

intptr\_23 = intptr\_0 + 0xD45A0;//1005 3EC 000D2FC0 mono\_type\_get\_object

intptr\_24 = intptr\_0 + 0x9E1E0;//695 2B6 0009D4E0 mono\_object\_get\_class

intptr\_25 = intptr\_0 + 0xA1190;//930 3A1 000A0480 mono\_string\_new

intptr\_26 = intptr\_0 + 0xA0150;//861 35C 0009F360 mono\_runtime\_invoke

intptr\_27 = intptr\_0 + 0xB3B90;//945 3B0 000B26F0 mono\_thread\_current

intptr\_28 = intptr\_0 + 0x32150;//115 72 00031D90 mono\_class\_get\_parent

intptr\_29 = intptr\_0 + 0x26410;//437 1B4 00026040 mono\_get\_root\_domain

intptr\_30 = intptr\_0 + 0x392C0;//298 129 00038DB0 mono\_event\_get\_name

intptr\_31 = intptr\_0 + 0xB35C0;//942 3AD 000B2120 mono\_thread\_attach

intptr\_32 = intptr\_0 + 0xA1770;//938 3A9 000A0A30 mono\_string\_to\_utf8

intptr\_33 = intptr\_0 + 0xA1A30;//949 3B4 000A0D20 mono\_thread\_get\_main

intptr\_34 = intptr\_0 + 0x99F90;//6 5 000992D0 mono\_array\_addr\_with\_size

intptr\_35 = intptr\_0 + 0x2E7D0;//9 8 0002E410 mono\_array\_element\_size

intptr\_36 = intptr\_0 + 0x9E260;//698 2B9 0009D560 mono\_object\_get\_virtual\_method

intptr\_111 = intptr\_0 + 0x9C590;//321 140 0009B8D0 mono\_field\_set\_value

intptr\_112 = intptr\_0 + 0x9C230;//319 13E 0009B500 mono\_field\_get\_value

intptr\_113 = intptr\_0 + 0x9C5E0;//322 141 0009B920 mono\_field\_static\_get\_value

intptr\_114 = intptr\_0 + 0x9C710;//323 142 0009BA50 mono\_field\_static\_set\_value

intptr\_115 = intptr\_0 + 0x9BA80;//147 92 0009ADD0 mono\_class\_vtable

intptr\_116 = intptr\_0 + 0x39390;//317 13C 00038E80 mono\_field\_get\_parent

2021-09-19 05:49:24,312 [BotThread] ERROR Logger (null) - [Tick] Exception during execution:

GreyMagic.InjectionDesyncException: Process must have frozen or gotten out of sync: Injection Finished Event was never fired

at GreyMagic.Executor.WaitForInjection(Int32 timeout)

at GreyMagic.Executor.SharedExecuteLogicEnd(Int32 timeout)

at GreyMagic.Executor.Execute(Int32 timeout)

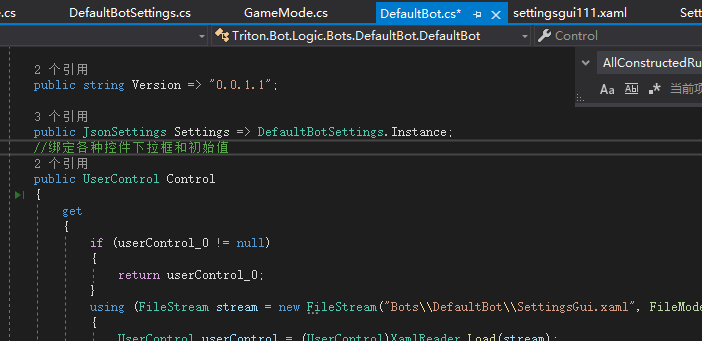
at GreyMagic.Executor.GrabFrame()

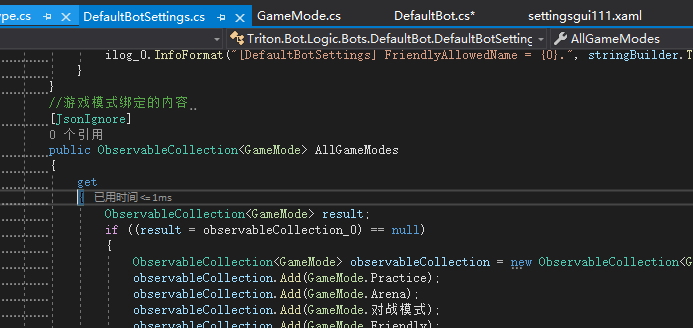
at GreyMagic.ExternalProcessMemory.AcquireFrame(Boolean isHardLock)

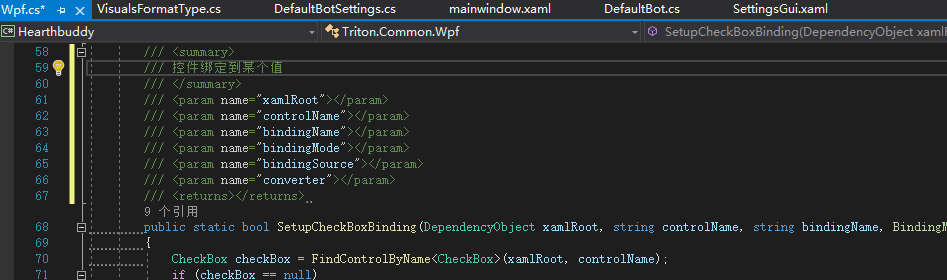
at Triton.Game.TritonHs.AcquireFrame()

at Triton.Bot.BotManager.smethod\_1(IBot ibot\_1)

因为炉石本身会冻结，没有问题







GraphicsResolution

IsAspectRatioWithinLimit

