MTGLDebug

MTGLDebug使用手册

- MTHawkeye提供对外接口, 和加载时机
- 对外唯一接口设置
 - MTGLDebugDisable 【需要重启App】完全关闭MTGLDebug不做hook对原本性能完全无影响
 - 。 MTGLDebugOptionsOnlyStatisticalGLObject【默认】 进行统计不做逻辑判断,不会有异常抛出 【addObject不做历史记录操作,check逻辑return,removeObject不做历史记录操作】 【优化点:仅对内存相关的GL函数进行hook】

对原本性能有稍微影响,如果只做性能检测,不做内存检测和错误检测建议用Disable

关闭MTGLDebugOptionsOnlyStatisticalGLObject执行

- MTGLDebugOptionCheckContextStates 【上下文资源逻辑检测】
 一般是图像会出错
- MTGLDebugOptionCheckAPIUsageStates 【glAPI检测】
 - 一般是崩溃在某个gl函数

案例演示

0x1 一些测试案例

0x2 美颜相机重复进出相机,拍摄界面卡死

卡死后控制台log

```
<[mlab] ERROR:> glerror = 506
```

开启GLDebug, 重复刚才的操作

```
● ● ● ■ ☑ MYXJ〉■ Zed
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       8 ( > 1 Pods) NotaryCrashReporter) NotaryCrashReporter) NotaryCrashReporter NotaryCras
                                                                                                                                          void __cxa_throw(void* thrown_exception, std::type_info* tinfo, void (*dest)(void*))
                                                                                                                                                      if(g_captureNextStackTrace)
               0 __pthread_kill
1 pthread_kill$VARIANT$mp
                                                                                                                                                                     g_stackTraceCount = backtrace((void**)g_stackTrace, sizeof(g_stackTrace) / sizeof(*g_stackTrace));
                                                                                                                                                    static cxa_throw_type orig_cxa_throw = NULL;
unlikely_if(orig_cxa_throw == NULL)
8 ::_cxa_throw(void *, std::type_info *, void (*)(void *))
                                                                                                                                                      {
    orig_cxa_throw = (cxa_throw_type) dlsym(RTLD_NEXT, "__cxa_throw");
          10 GLDebug::MTGLDebugCore::checkObject(GLDe
           11 decitype(std::_1::forward<GLDebug::MTGLDebugC...
12 void std::_1::_invoke_void_return_wrapper<void>::_...
                                                                                                                                                     orig_cxa_throw(thrown_exception, tinfo, dest);
__builtin_unreachable();
                                                                                                                                                                                                                                                                                                                                                                                                                                 com.meitu.MTGLDebug (11): signal SIGABRT
           13 std::_1::_function::_func<GLDebug::MTGLDebugC...
14 std::_1::function<void ()>::operator()() const
          18 void std::_1::_thread_execute<std::_1::unique_ptr...
19 void* std::_1::_thread_proxy<std::_1::tuple<std::_....
                                                                                                              101 static void CPPExceptionTerminate(void)
102 {
                                                                                                                                        KSLOG_DEBUG(@"Trapped c++ exception");
   ► ① Thread 12
► ① Thread 13
► ① WebThread (14)
► ① Thread 15
                                                                                                                                       bool isNSException = false;
char descriptionBuff[DESCRIPTION_BUFFER_LENGTH];
const char* name = NULL;
const char* description = NULL;
   2018-03-12 20:02:14.357509+0800 MYXJ[1207:645401] Created new private queue context: <NSManagedObjectContext: 0x1c41d5450>
2018-03-12 20:02:14.537340+0800 MYXJ[1207:645031] [] <<<< AVOutputDeviceDiscoverySession (FigRouteDiscoverer) >>>> -
[AVFigRouteDiscovererOutputDeviceDiscoverySessionImpl outputDeviceDiscoverySessionDidChangeDiscoveryMode:]: Setting device
discovery mode to DiscoveryMode.None (client: MYXJ)
2018-03-12 20:02:14.576294+0800 MYXJ[1207:645031] [Default] MPMediaControlsRemoteViewController Dismissing because view
                                                                                                           libc++abi.dylib: terminated libc++abi.dylib: terminating with uncaught exception of type std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         11 1
```

我们就挂在GLDebug抛出的异常上

```
Pods Develop
► M Thread 3
                                                                                                                                                                                  case MTGLDebugObjectStatusUseWronglyOperateGLObject:
                                                                                                                                                                                                        debugObject.debugDescription();
                                                                                                                                                                                 break;
case MTGLDebugObjectStatusUseForwardlyDeletedGLObject:
{
                                                                                                                                                                                                errorString = "当前上下文不包含这个对象!!!!!!,在当前上下文创建过但是被销毁,使用已被删除的对象" +
                                                                                                                                                                                                          debugObject.debugDescription();
      ### 10 (Debtoy MTG Debtoy on the Object (Clubboy MTG Debtoy Control of Clubboy Control 
                                                                                                                                                                                                                                                                                                                                                                                                                                  = com.meitu.MTGLDebug (11): signal SIGABRT
                                                                                                                                       });
≥ 22 thread_star

► ① Thread 12

► ① Thread 13

► ② WebThread (14)

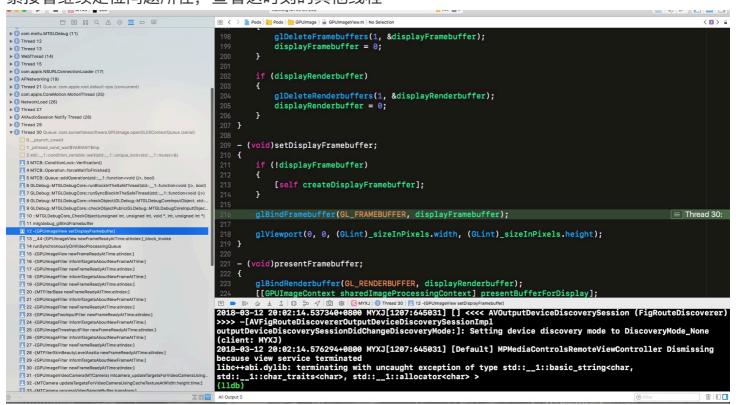
► ② Thread 15
                                                                                                                                       return result;
 ► ( com.apple.NSURLCo

► ( AFNetworking (19)
 ► (1) Thread 21 Queue
                                                      nThread (25)
                                                                                                            2018-03-12 20:02:14.357509+0800 MYXJ[1207:645401] Created new private queue context: <NSManagedObjectContext: 0x1c41d5450>
2018-03-12 20:02:14.537340+0800 MYXJ[1207:645031] [] <<<< AVOutputDeviceDiscoverySession (FigRouteDiscoverer) >>>> -
 ▶ ( NetworkLoad (26)
 ► ( Thread 27
                                                                                                         [AVFigRouteDiscovererOutputDeviceDiscoverySessionImpl outputDeviceDiscoverySessionDidChangeDiscoveryMode:]: Setting device discovery mode to DiscoveryMode_None (client: MYXJ) 2018-83-12 20:02:14.576294+0800 MYXJ[1207:645031] [Default] MPMediaControlsRemoteViewController Dismissing because view
 ► ① Thread 29
► ① Thread 30
 ► ① Thread 32
► ① Thread 33
                                                                                                         2010 of 12 E-0012...
service terminated
libc++abi.dylib: terminating with uncaught exception of type std::__1::basic_string<char, std::__1::char_traits<char>,
 com.apple
                                                                                                         std::_1::allocator<char> > (lldb)
```

左侧的堆栈往上找我们会看到com.meitu.MTGLDebug线程给出的错误提示

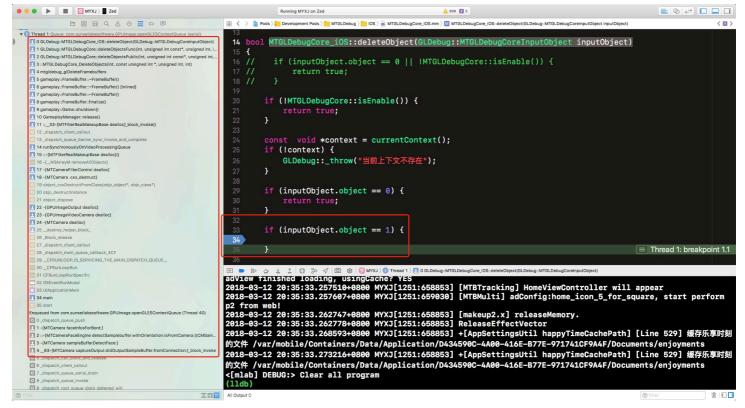
⚠当前上下文不包含这个对象,使用了被删除的对象

紧接着继续定位问题所在, 查看这时刻的其他线程

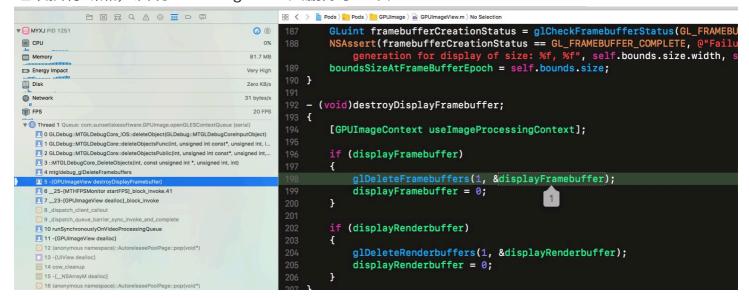


从图中我们可以看出是运行glBindFramebuffer这个函数检测GLDebugObject的时候出的问题。

然后,从上面抛出的异常信息可以判断,setDisplayFramebuffer目前绑定的displayFramebuffer 其实已经被删除,所以导致拍摄画面不会被更新,显示出画面卡在上次退出的画面上。 接着验证,如果多删除了一次,在deleteFramebuffer这个函数删除两次句柄为 displayFramebuffer的 Fbo。由上面得出displayFramebuffer为1,所以在delete函数加上 inputObject==1 的判断。这里 MTFilterRealMakeupBase里面的 gamePlay::Frambuffer 删除了 这个displayFramebuffer 有异常



继续执行断点、发现GPUImageView又删除了一次



MTGLDebug设计说明

- 创建DebugObject对象
 - 。 addObject逻辑

- 取当前的上下文检测是否存在
- 以sharegroup地址为key值,共享的GL资源为value 【创建sharegroups字典】
- 一个记录当前,一个记录历史
- 如果sharegroup不在sharegroups中则以地址为key值加入sharegroup为value加入 到sharegroups
- 根据gl对象的target和object组合的字符串作为key值在加载入sharegroup共享资源字典里
- 。 CheckObject逻辑



- DeletelObject逻辑
 - 删除当前,历史记录不删

附录:

用同一个shareGroup创建Context,才能资源共享

```
_movieWriterContext = [[GPUImageContext alloc] init];
[_movieWriterContext useSharegroup:[[[GPUImageContext sharedImageProcessingContext] context] sharegroup];

- (void)useSharegroup:(EAGLSharegroup *)sharegroup;
{
    NSAssert(_context == nil, @"Unable to use a share group when the context has already been created. Call this method before you use the context for the first time.");
    _sharegroup = sharegroup;
}

- (EAGLContext *)createContext;
{
    EAGLContext *context = [[EAGLContext alloc] initWithAPI:kEAGLRenderingAPIOpenGLES2 sharegroup:_sharegroup];
    NSAssert(context != nil, @"Unable to create an OpenGL ES 2.0 context. The GPUImage framework requires OpenGL ES 2.0 support to work.");
    return context;
}
```

如果是要另一份不共享shareGroup就填nil去创建上下文