# KSVG-EDIT程序-内存泄漏分析报告

【摘要】

本报告采用VMMap内存分析工具、VLD内存泄露分析工具、UMDH内存占用分析工具以及VS内置诊断工具，对**KSVG-EDIT程序**的内存占用情况、内存泄露问题以及内存优化方案进行说明。报告中将对各个分析工具的使用进行说明，分析以降低特定功能的内存使用量、排查内存占用异常上涨为目的，通过中断分析调用栈、记录内存分配比较差异的方法来进行。

【关键词】

内存泄漏、内存优化

**一、项目内存占比分析**

VMMap是对进程所用虚拟、物理内存进行分析的工具，可以统计进程所使用的虚拟、物理内存并以条形图与文字的形式进行呈现。VMMap的过滤和刷新功能可以及时观察进程内存使用的来源以及占用情况。

运行SVG编辑器项目，并使用VMMap工具进行检测，可首先观察到如下界面，图1.1。

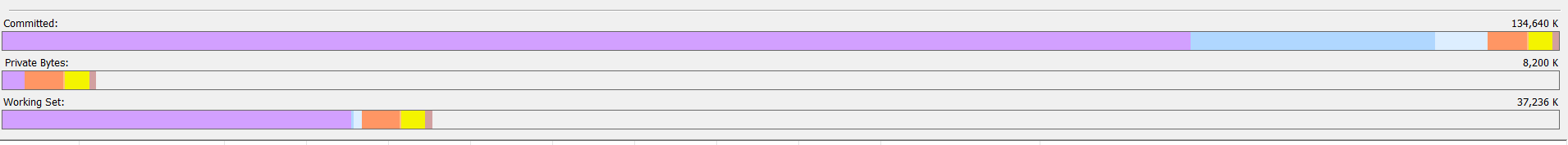
程序**启动后**，进程内存总体信息状态如下：

图1.1 VMMap中进程内存总体情况（启动后）

其中顶部为进程基本信息，下部为各类类型空间占用条形图。Committed表示进程向操作系统所申请的需要使用到的虚拟内存大小；Private Bytes表示进程所私用使用的虚拟内存大小；Working Set表示进程目前所使用的物理内存大小。

从图中可知，程序**启动后**，虚拟内存分配空间大小为134,640K，项目所私有使用的虚拟内存空间大小为8,200K，进程所实际使用的物理内存空间大小为37,236K。

在工具的中部区域，对进程内存总体信息进行了进一步分析

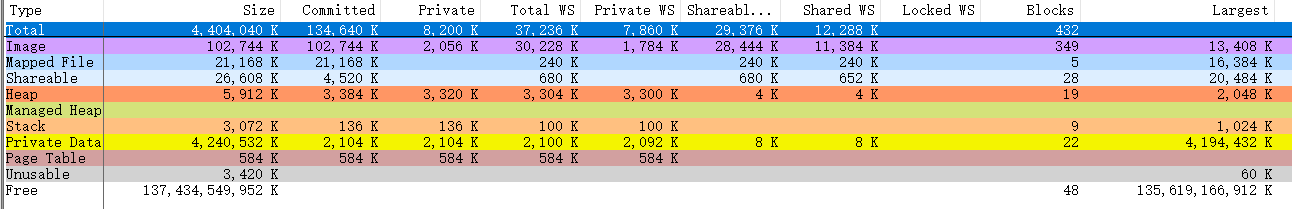


图1.2 VMMap中进程内存总体情况次详细信息（启动后）

横向表头分别代表总体内存大小、申请内存大小、私有内存大小，总体物理内存带下、私有物理内存大小、共享物理内存大小等类型。纵向表头代表总体、进程自身可执行文件以及包含的所有库、内存映射文件、共享内存、堆区空间、栈区空间等类型。

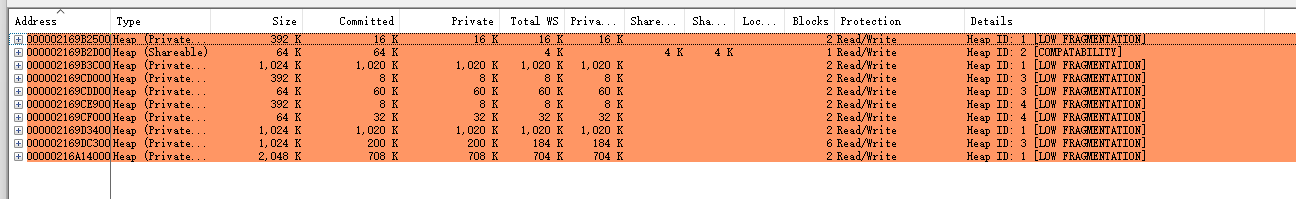


图1.8 VMMap中进程堆区空间状态（启动后）

程序运行一段时间后，进程内存总体信息状态如下：

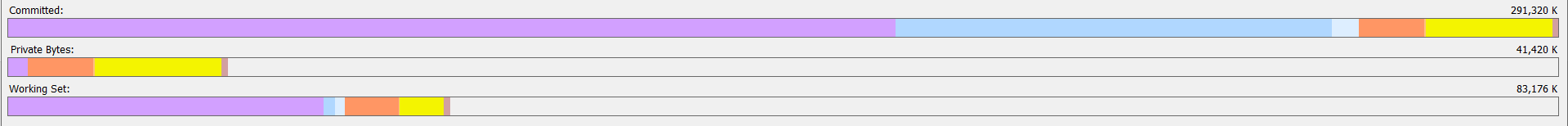


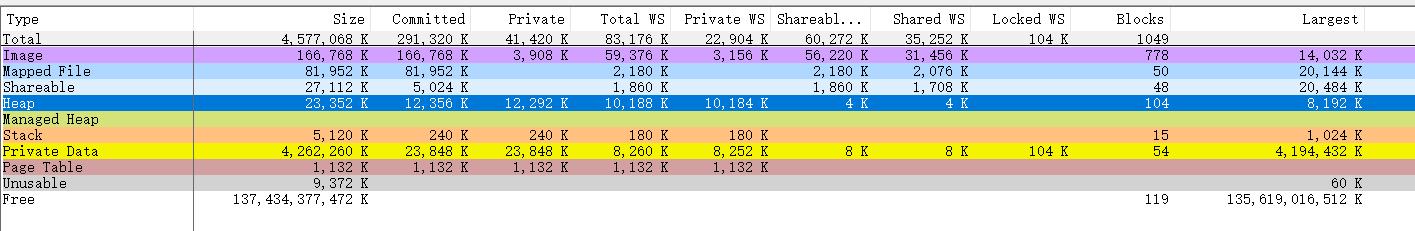
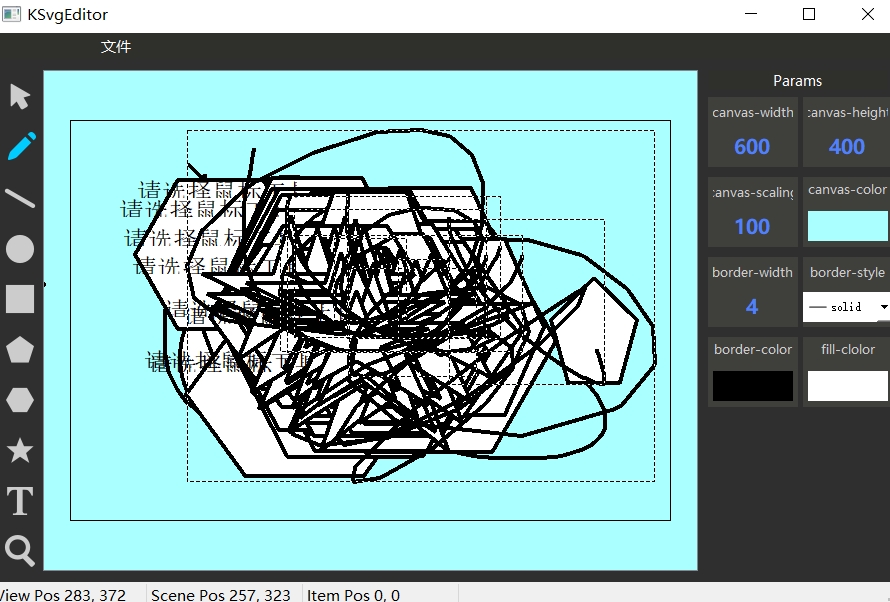
图1.3 VMMap中进程内存总体情况

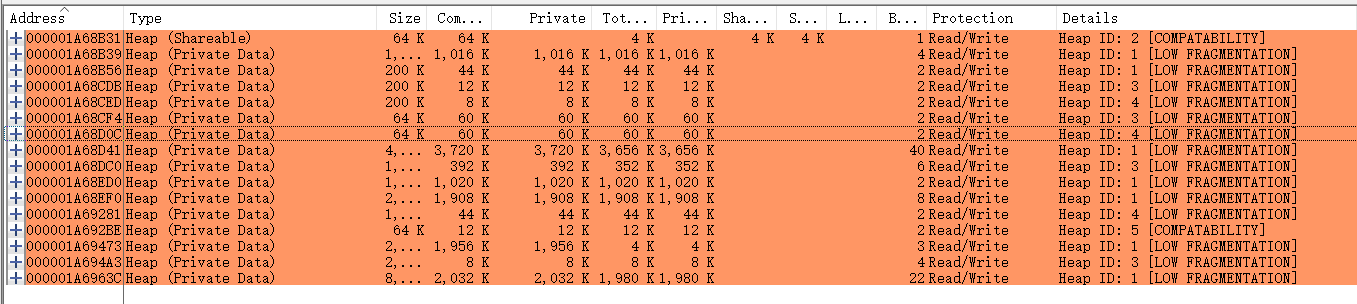
图1.4 VMMap中进程内存总体情况次详细信息

从图中可知，程序执行时，虚拟内存分配空间大小为291,320K，项目所私有使用的虚拟内存空间大小为41,420K，进程所实际使用的物理内存空间大小为83,176K。



其中new与malloc等分配的空间为堆区域，栈区域空间会主动释放，因此进一步对堆区空间进行分析。

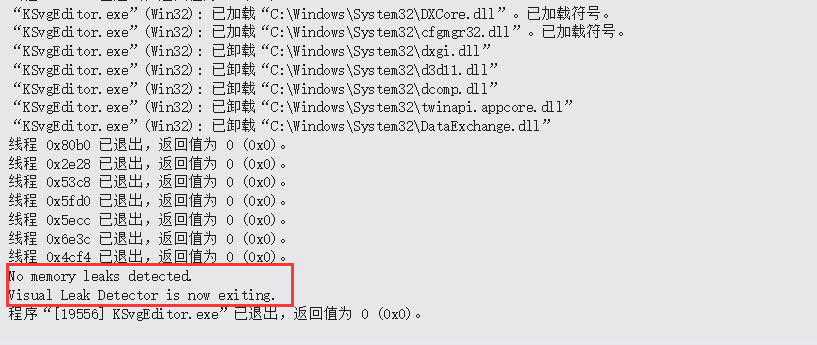
程序绘制大量图形后，堆区空间状态如下：



**二、项目内存泄露点**

 Visual Leak Detector，它是一款用于 Visual C++免费的内存检测工具。VLD可以得到内存泄漏点的调用堆栈、泄露内存的完整数据，还可以设置内存泄露报告的级别。在VS项目中，加上<vid.h>头文件后运行一次程序，即可得到内存泄露分析结果。可以**检测出疑似内存泄露**的具体行号，new生成的对象使用完毕后未及时释放，因此发生内存泄露。

**内存泄露检测结果：**

**程序未出现内存泄露现象**

**除了图形对象手动释放以外，其余全都使用Qt的对象析构树进行对象的析构与释放**

**三、UMDH的介绍**

UMDH（user Mode dump heap)是 Debugging Tools for Windows中附带的一个工具。该工具可以分析当前进程在堆上分配的内存，通过命令操作，可以对当前进程分配的每一块内存做日志记录。

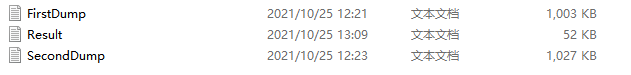


图3.1 UMDH中项目内存分析

**四、VS诊断工具的使用与项目内存分析**

在VS中，可启用自带的诊断工具对**程序**项目内存占用情况进行监控与分析，如图4.1。

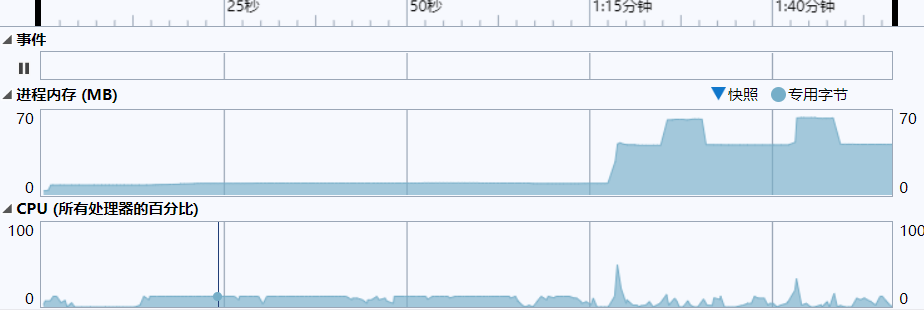


图4.1 VS诊断工具中项目内存分析

所有的内存占用飙升均为保存为SVG文件，之后会恢复到稳定状态

**五、项目内存优化**

QGraphicsView底层使用树数据结构对图形对象进行管理，因此效率很高，可以快速获取某一点的图形，但是有一个注意的点就是该框架的视图在删除一个图形后并不会释放其空间，而是需要我们手动释放

在绘制图形时，只有在创建一个图形时创建图形对象，之后的各种操作均设置图形rect、pen和brush进行重绘，避免过多的对象创建与析构过程

尽量使用qt的对象树来进行对象的析构删除

传参时要注意对象的拷贝，必要时可以使用引用或者右值传递，避免开销

在绘制自由线时候会保存大量坐标点，可以采取点的压缩算法进行优化