# 一、项目经验

## <1>关于流媒体开发

### －需求分析

## <2>物体位置信息以及标牌避让

### －需求分析

|  |  |  |
| --- | --- | --- |
| **具体需求** | | **性能要求** |
| 600移动的物体及其轨迹展示在坐标轴上 | | 不卡顿 |
| 与600移动的物体(可以看成一个移动的点)一一对应相连的600个标牌以及连线 | |
| 标牌与运动点之间交互关系的处理 | 双击标牌可以对运动点的信息进行显示 |
| 选中或者拖拽标牌（标牌与运动点的连线可以实时改变） |
| 右击标牌弹出菜单可以对运动点进行设置（比如让运动点进行居中显示） |
| 600个标牌需要进行避让 | |

### －重构原因

１：QWT的限制

２：新性能以及新需求的要求

### －难点攻破

１：性能不足

|  |  |  |
| --- | --- | --- |
| **渲染方法** | **优点** | **缺点** |
| QWT |  |  |
| paintEvent |  |  |
| Label->setPixMap |  |  |
| 其他 |  |  |

２：标牌避让

### －收获

## <３>框架搭建

功能：

## <４>Ｃ＋＋编码规范

### －变量初始化

一定要注意变量初始化，否则容易出现一些意想不到的问题。

指针初始化未指空或者delete(free)为指空，导致野指针的问题。

用于控制代码流向的变量未初始化导致，代码流向不确定。

int flag;

if( flag >= 1 )

{

//Do something;

}

else if

{

//Do something;

}

其中vs的debug会给代码初始化，而release则不会容易导致问题出现。

为避免变量未初始化一定要主意以下地方

**普通变量声明的时候便初始化**

**成员变量一定要在构造函数将其初始化**

**结构体要带有默认的构造函数**

### －容器越界

使用容器一定要主意保护边界，否则很容易造成崩溃。

访问map之前一定要判断其是否有该key值

访问list或者vector一定要其边界，是否为空？是否访问了容器最大值之外的值。

如果容器可能会因为程序运行变得越来越大，一定要设置极限值，当超越该值的时候及时出该容器。

### －代码冗余

无效代码，伴随者程序开发一定会出现一些无效的变量或者无效的逻辑代码，一定要及时**注释甚至删除**，否则会导致后续代码不易维护，以及增加二次开发时，代码解读的难度。

重复代码，相同的代码逻辑可以将其抽成函数，结构一样的代码可以用模板。

### －代码重构

## <５>Ｃ＋＋的bug调试经验