Gui主线程完全是通过事件调度器去实现，我们需要一个事件调度器完成该目标

**设计参考**：

他是一个以 单/双链表 为容器的 队列/优先队列，使用互斥锁完成多线程的同步和事件队列的保护，事件入队列的时候要产生一个信号量，该信号量将在未来激活Gui主调度线程去处理该事件

你的事件管理器不仅仅要事件事件入队列，优先级入队列，相同的事件需要适当的合并，特定的事件需要命中移除

同样的，事件管理器还需要生成为特定事件挂载响应回调的容器，因为控件去响应事件时，可以优先响应特定事件的专属响应回调，没有之后再去走通用事件响应回调，这样子可以让你在ui层有更加灵活的组合处理特定事件的手段

**结构参考**：

你的事件应该包含以下的数据类型：

事件对象（系统事件，或者某一个控件（或控件树）的事件）

事件的优先级（决定事件在队列中的排序情况（优先队列））

事件是否同步调度（就地响应，不入事件队列），或者异步调度（入事件队列，产生信号量，在之后的调度周期响应）

**例**：额外扩展的数据类型，可以用一个匿名联合体包裹结构体标明，

TP事件有以下的类别（down，up通常需要携带屏幕的报点，click通常需要额外携带连续按下次数，move，fling通常需要额外携带屏幕的路径，比如说从a点滑动到b点，hold通常需要额外携带从down产生到当次hold中经历的事件）

编码器事件有以下类型（正转与反转，以及幅度）

按键事件有以下类型（down，up有按键号和按键值，hold有额外持续时间，click有按键额外连续按下次数）

/\*@brief 事件包入列函数

\*@param event 事件包

\*/

void scui\_event\_enqueue(scui\_event\_t \*event);

/\*@brief 事件包出列函数

\*@param event 事件包

\*@retval 提取到有效事件包

\*/

bool scui\_event\_dequeue(scui\_event\_t \*event, bool hit);

事件响应流程中，事件响应分为多个步调去执行，不过大部分事件只需要一个步调，这是一个特殊的事件，绘制事件，绘制事件完成向画布绘制特定的图形，该事件通常有三个步调，从prepare->execute->finish三个步调，如果是小内存分段刷机制，则prepare承担了为绘制动作前的一些绘制信息的全局计算，特定内存空间的开辟，甚至是特定资源的准备，然后finish步调中回收资源等工作

/\*@brief 事件回调列表事件查找

\*@param cb\_list 事件回调列表

\*@param cb\_node 事件回调节点

\*/

void scui\_event\_cb\_find(scui\_event\_cb\_list\_t \*cb\_list, scui\_event\_cb\_node\_t \*cb\_node);

/\*@brief 事件回调列表事件添加

\*@param cb\_list 事件回调列表

\*@param cb\_node 事件回调节点

\*/

void scui\_event\_cb\_add(scui\_event\_cb\_list\_t \*cb\_list, scui\_event\_cb\_node\_t \*cb\_node);

/\*@brief 事件回调列表事件移除

\*@param cb\_list 事件回调列表

\*@param cb\_node 事件回调节点

\*/

void scui\_event\_cb\_del(scui\_event\_cb\_list\_t \*cb\_list, scui\_event\_cb\_node\_t \*cb\_node);