1.Layer

virtual void OnEvent(Event& event){}

2.Layerstack

std::vector<Layer\*> m\_Layers;

3.在application里

LayerStack m\_LayerStack;

在这添加层

void Application::PushLayer(Layer\* layer) {

m\_LayerStack.PushLayer(layer);

layer->OnAttach();

}

void Application::PushOverlay(Layer\* layer) {

m\_LayerStack.PushOverlay(layer);

layer->OnAttach();

}

在event函数中，循环调用每个layer的自己的事件event

EventDispatcher dispatcher(e);

dispatcher.Dispatch<WindowCloseEvent>(BIND\_EVENT\_FN(OnWindowClose));// OnWindowClose是本类的方法，用来取消游戏循环的

// 从后往前

for (auto it = m\_LayerStack.end(); it != m\_LayerStack.begin();) {

(\*--it)->OnEvent(e);

if (e.Handled)

break;

}

比如：

Imgui的event

在sandboxapp中，添加到application的layerstack中

PushOverlay(new Hazel::ImGuiLayer());

，然后循环调用application的event调用imgui的event

注意：imgui的event不是打印事件，而是响应自己ui的一些事件

解释：为什么examplelayer能接受打印窗口事件

由笔记1得，application指定了m\_Window->SetEventCallback(BIND\_EVENT\_FN(OnEvent));窗口事件绑定application的Onevent事件

当窗口执行事件：data.EventCallback(event);，则执行application的OnEvent

这里就是循环各个layer，然后执行onevent，而传的参数就是e，这个e就是窗口事件！