Chapter 14: Handling OS 🖱 InputEvents 🕹

we are gonna take an backwards unfolding approach here

1. 📨 dispatch_events

- `amGHOST` ***StackTrace*** *[Surface ---> Deep]*

```
amGHOST:- lets your headache dissappear about OS::Stuffs/Functions

1. amGHOST_Window     ::dispatch_events_with_OSModalLoops()     🔗[github][1]
2. amGHOST_WindowWIN32::dispatch_events_with_OSModalLoops()     🔗[github][2]
3. amGHOST_System     ::dispatch_events_with_OSModalLoops()     🔗[github][3]
4. amGHOST_SystemWIN32::dispatch_events_with_OSModalLoops()     🔗[github][4]
5. amGHOST_SystemWIN32.cpp::actual_implementaion                🔗[github][5]
```

  - *1. amGHOST_Window*
  - *2. amGHOST_WindowWIN32*
  - *3. amGHOST_System*
  - *4. amGHOST_SystemWIN32*
  - *5. amGHOST_SystemWIN32.cpp*

- ***[win32] it has 6 parts***
  - *The plan is to get this section generalized for all OS & have separate* `win32` / `xlib` / `x11` / `wayland` / `macOS` *sections below*

```
1. ::PeekMessage() or ::GetMessage()
2. ::TranslateMessage()
3. ::DispatchMessage() ---->   WndProc()
4. static WndProc() -------> ::DefWindowProcessor()
5. ::DefWindowProcessor() -> ModalLoop
6. ModalLoop
```

vii. `::PeekMessage()`
  - *Kinda like pop & grab the last InputEvent/Message from EventQueue*

viii. `::TranslateMessage()`
  - *This is needed on some OS, for some specific ops, e.g. KeyBoard events*

ix. `::DispatchMessage()` --> `WndProc`
  - *For now, i only know about one kind of Dispatching*

  a. *Calls* **WindowProcessorFunction** / `WndProc`
    - *must have been registered & tired to a window duriing WindowCreationg*

x. `static WndProc()`
  - *InputEvent Processor as per Window*

xi. `::DefWindowProcessor()`
  - *Operating System's very own little* **InputEvent** *Processor*
  - *Properly Unhandled events must be passed on to this function*
  - *Some InputEvents/Messages*
    - *can't really be Properly Handled, e.g.* `WM_SYSCOMMAND`
  - *Get's Into* **ModalLoops** *during events like WINDOW-RESIZING*

xii. *Modal Loops*
  - *See below* 👨‍🏫

2. `win32` 📄

1. ***General Info*** ℹ
   i. `::CreateWindowA()`
      - *Every* **win32 window** *needs a* `WNDCLASSA` *during* `::CreateWindowA()` ⏳ |
      - *Window gets* **bound** *to the* **calling/creating thread** *as* **owner thread**
   ii. `::Peek/Get/DispatchMessage()`
      - *does not peek/get/dispatch messages of windows from other threads*
   iii. `WNDCLASSA.lpfnWndProc` 🔗
      - *Binds the window to a* `static WndProc()` ⚙️
   iv. `static WndProc()`
      - WindowProcessorFunction
      - *This is a function that you need to implement.*
      - *It needs to be* `static`
      - ⚠️ Beware of Static Initialization Order Fiasco
        - 📺 *50-min CppCon Talk*
        - ⏳ *12-min explainer*
        - 🔍 *Extras: 1*
        - 🎥 *Extras: Singleton pitfalls*

---

3. ModalLoop 🔄

- ***I: Core Behavior*** ⚙️
  - *Starts @* `MouseButtonDown` 🖱️
  - **Peeks**, **Dispatches** *(OtherMessages)*, **Waits** *till* `MouseButtonReleased`
  - *Ends @* `MouseButtonReleased` 🖱️
- ***II: Triggered During*** 🎬 ***events like***
  - ReSizing ↔️
  - *Minimize/Maximize* **animations** *(Windows Aero effects)* 🪟
- ***III: Halts / Blocks*** `Thread`

ModalLoop 🔄 : `win32`

- ***IV:*** `ModalLoop` ***Under The Hood:-***
  - *OS has it's own little* `dispatch_events()` *implementation.*
  - *It keeps on* **peeking** & **dispatching** & **waiting** *till* `MouseButtonReleased` *is received*

```
while(true) {
    ::GetMessage()          // Blocks Thread if no message is there till smth arrives
    ::TranslateMessage()
    ::DispatchMessage()
        calls -> static WndProc()      [userProvided]

    if (msg == WM_EXITSIZEMOVE) {break;}
        // Obviously there will be a heck ton other extra processing going on, but you get the idea
}
```

  - *It's so darn similar to* MainLoop

- *V:-* `win32` *Sample Implementation (odin32):-*
  i. ➡️ *win32wbase.cpp#L1581*
  ii. ➡️ *win32wbase.cpp#L1922*
  iii. ➡️ *win32wbasenonclient.cpp#L1318*
  iv. *SC_SIZE: This is the key for Window Resizing event*
  v. ➡️ *wintrack.cpp#L441*
  vi. *& Here goes the ModalLoop*
  vii. ➡️ *wintrack.cpp#L564*

1. *FAQ* ❓
   - Exactly where does the ModalLoop gets Trigger? 🎬
     - *When we pass* `WM_SYSCOMMAND` *to* `::DefWindowProc()`
   - `WM_LBUTTONDOWN` *vs* `WM_NCLBUTTONDOWN` 🖱️
     - *Pressing mouse on* OS-Window Frame/Corner/Border ➡️ *sends* `WM_NCLBUTTONDOWN` *(not* `WM_LBUTTONDOWN`*)*
     - NC *= Non-Client* 🏷️
     - LBUTTON *= Left Mouse Button*
   - What if we ignore `WM_NCLBUTTONDOWN`? ⛔
     - `WM_SYSCOMMAND` *won't generate!*
     - Must call `::DefWindowProc()` on `WM_NCLBUTTONDOWN` ☑️
     - *Passing This one to* `::DefWindowProc()` *is exactly how the OS internally keeps track of* MouseButtonDown 🕵️
   - When does `WM_ENTERSIZEMOVE` occur? 🔄
     - *When you call* `::DefWindowProc()` *with* `WM_SYSCOMMAND`
   - ModalLoop starts on passing `WM_SYSCOMMAND` or `WM_ENTERSIZEMOVE`? 🤔
     - *I believe, it's* `WM_SYSCOMMAND` *[gotta test* 🧪*]*
   - If `WM_SYSCOMMAND` starts the `ModalLoop`. why'd we even catch `WM_ENTERSIZEMOVE` in our `static WndProc()`?
     - *Well, it's because,* `win32` *wanted us to catch all those events, but still wanted us to call* `::DefWindowProc()` *on those*

ModalLoop 🔄 : `xlib`

- *IV:* `X11` / `XCB` / `Wayland` *[Linux] & [MacOS]*
  - *To be added, really soon! (when i port amGHOST to x11/xcb/wayland/macOS)*

ModalLoop 🔄 : `summary`

- *VI: In Words:-*
  - *Operating System enters it's very own* `Loop` *when* `MouseButton` *is* PressedDown
  - *This is what's called* `ModalLoop`
  - *The* `ModalLoop` *doesn't* `return/break` *till* `MouseButton` *is* Released
    - *Yes, that does mean that your* mainThread is blocked and waiting!

4. Resizing ↔

- ***Generates*** `WM_SYSCOMMAND`
  - *right when we pass* `WM_NCLBUTTONDOWN` --> `::DefWindowProcessor()`
- ***Triggers ModalLoop***
  - *right when we pass* `WM_SYSCOMMAND` --> `::DefWindowProcessor()`
- ***Messages sent while inside ModalLoop***

```
[REPEATED]:- WM_NCMOUSEMOVE --> WM_NCHITTEST --> WM_SETCURSOR
[REY_MODAL_LOOP]:- WM_NCLBUTTONDOWN  --> Entering: DefWindowProc
[REY_MODAL_LOOP]:- WM_SYSCOMMAND  --> Entering: DefWindowProc
    [Win32GUI]:- WM_GETMINMAXINFO
    [Win32GUI]:- WM_ENTERSIZEMOVE
    [Win32GUI]:- WM_NCMOUSELEAVE
    [Win32GUI]:- WM_CAPTURECHANGED
    [Win32GUI]:- WM_WINDOWPOSCHANGING
    [Win32GUI]:- WM_GETMINMAXINFO
    [Win32GUI]:- WM_EXITSIZEMOVE
[REY_MODAL_LOOP]:- WM_SYSCOMMAND  --> Returned: DefWindowProc
[REY_MODAL_LOOP]:- WM_NCLBUTTONDOWN  --> Returned: DefWindowProc
[REPEATED]:- WM_NCMOUSEMOVE --> WM_NCHITTEST --> WM_SETCURSOR
```

- ***📜 REY_DOCs***
  - *Well, if you wanna do anything inside the Window, while it's being resized, You must do it in a different thread 👩‍🏫 because of the* **MainThread** *being stuck in the* `modalLoop`

5. Window Creation & Destruction

- *TBA: **WM_CREATE**, **WM_DESTROY**, **WM_KEYDOWN**, + Show Code / Redirect to my WndProc implementations*

1. *amGHOST Events <-- (Win32/XCB/X11/Wayland/macOS)*
   i. *EventTypes*
      - *https://www.youtube.com/watch?v=xnopUoZbMEk&list=PLlrATfBNZ98dC-V-N3m0Go4deliWHPFwT*