Page No.:

Assignment - 1.
"Essay Writing"

Name : Vipul Mukund Newlaskar.
Roll No. : WinRT (2019) - 067

Subject : Inside Message Loop.

Inside Message Loop ...

Win32 चं Message Loop यावर आपण आता लक्ष केंद्रीत करुयात. अहमदी Win32 च कशाला सर्वच SDKs मध्ये असं loop असणारे, फक्त ते वापरायच्या अथवा Handle करायच्या पद्धती different असतील.

आता वळूयात आपल्या विषयाकडे. समजा असे मानू, आपलं Application, Rather आपल्या Application चे २-3 instances running आहेत. या दोन्ही-तिन्ही instances ला स्वतःचा hinstance OS ने दिलाय. आणि, Applications च्या आपापल्या windows (os नाही) "Desktop" नावाच्या window "वर" (अहं, वर, खाली असं काहीच नाही) दिसत आहेत. आणि "focus" आपल्या window वर आहे.

आता समजा कोणीतरी Mouse ने आपल्या window वर click केलंय. त्याचा hardware interrupt साला, त्याला OS ने event, म्हणून consider केला, त्याला message मध्ये translate करुन application ला पाठवला (आपल्या application च्या त्या window handle ला ज्याच्यावर click झाला आहे)

आपल्या Application ची window तोपर्यंत चालू राहणार आहे, जोपर्यंत Message Loop break होत नाही.

आपलं Message Loop पुढीलप्रमाणे आहे:

```
:
//WinMain चा code
:
// Message Loop.
while (GetMessage(&msg, NULL, 0, 0
{
  :
}
```

थोडक्यात काय तर GetMessage जोपर्यंत false म्हणजे 0 return करत नाही तोपर्यंत आपलं application जिवंत आहे.

आता जरा GetMessage च्या parameter बद्दल

① &msg : msg चा address.
इथे आपण msg ला initialize केले नाही. ते आपल्याला OS परत देते.

msg चे 6 contents :
(i) Window handler.
(ii) event कुठला ते (उदा. WM_LBUTTONDO
(iii) WPARAM
(iv) LPARAM (co-ordinates वगैरे)
(v) system time.
(vi)

② HWND : कुठल्या Window चे messages handle करायचे ते. NULL म्हणजे आपल्या hInstance च्या सर्व Windows चे messages.

③ 0 –
④ 0 – } कुठल्या "range" मधले messages handle करायचे.

शक्यतो हे 3 parameters NULL, 0, 0 च द्यायचे

GetMessage च्या नंतर त्याची return value true असेल तर आता while च्या आतमध्ये.

आता TranslateMessage (& msg).
जर message Translate करायची गरज असेल तर msg translate होऊन तिथेच पुन्हा save होते.

DispatchMessage (& msg): आलेला msg Translate झाल्यावर DispatchMessage आपल्या application च्या event handler ला म्हणजेच WndProc ला call करतो.
Finally, WndProc मध्ये आपण switch case... break मध्ये आपलं application event handle करतो.
Ultimately हे सर्व events OS कडे by default divert केले जातात.

आता समजा आपण window चं close button press केलं अथवा कुठल्या ना कुठल्या प्रकारे close केलं तर आलेला msg WM_LBUTTONDOWN or WM_KEYDOWN म्हणून येतो. तो Translate होऊन WM_DESTROY होतो आणि WndProc मधल्या WM_DESROY ची case run होऊन PostQuitMessage(0) आल्यावर आपलं application end होतं.