

# API Windows

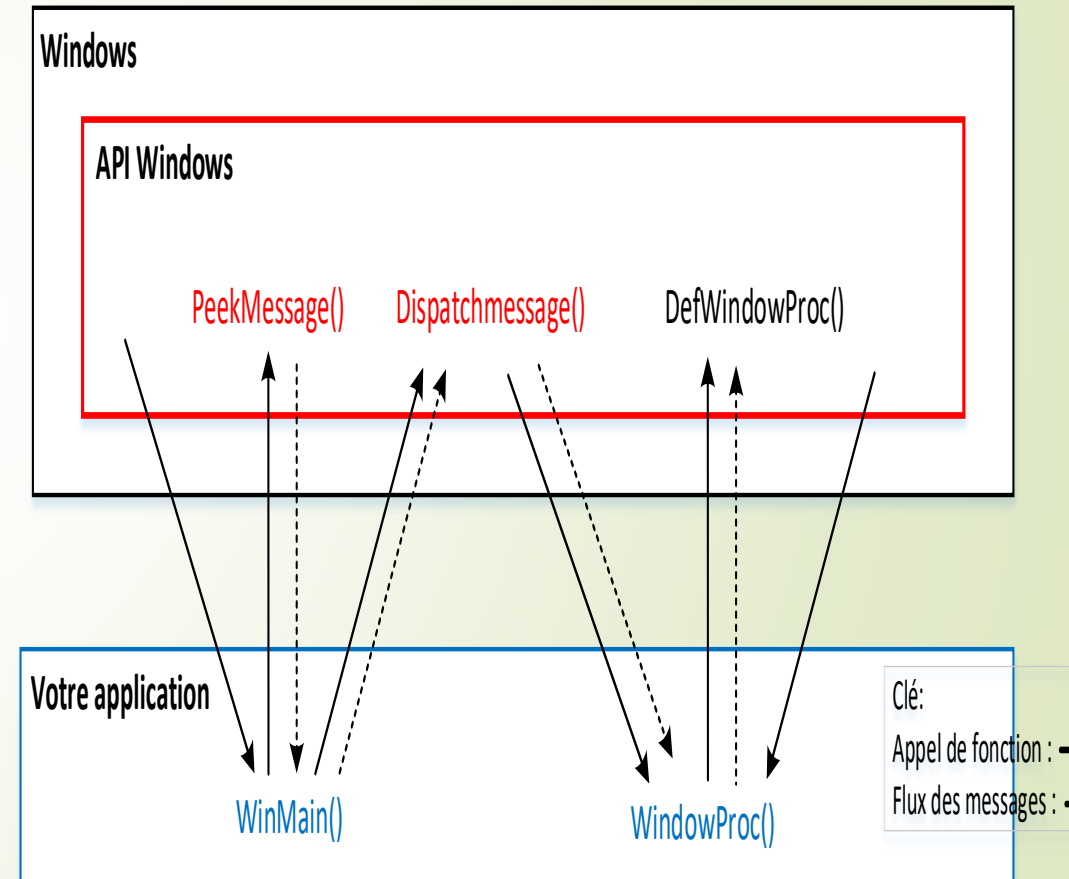
- La communication entre une application Windows (votre programme) et le système d'exploitation Windows s'effectue à l'aide de l'API Windows:
  - Cette API est constituée de milliers de fonctions dont les déclarations sont contenues dans le le fichier d'entête `<windows.h>`
- Les programmes Windows sont événementiels:
  - La majorité du code requis pour votre application est dédiée pour gérer les événements de type click-souris, bouton, appuyez sur une touche de clavier, etc.
  - Windows enregistre chaque événement dans une file d'attente de messages,
  - Chaque événement est enregistré dans un message contenant des information sur la nature de l'événement , son origine, le moment de sa génération.
  - Windows peut envoyer un message à votre application pour lui signifier d'exécuter une action afin de répondre à l'événement.
- Un programme appelle la fonction `PeekMessage()` de l'API pour obtenir un message de la file et renvoie « rapidement » le message à Windows en appelant une autre fonction `DispatchMessage()` qui demande à Windows d'envoyer le message au gestionnaire de la fenêtre (`WinProc()`).

# API Windows (2)

2

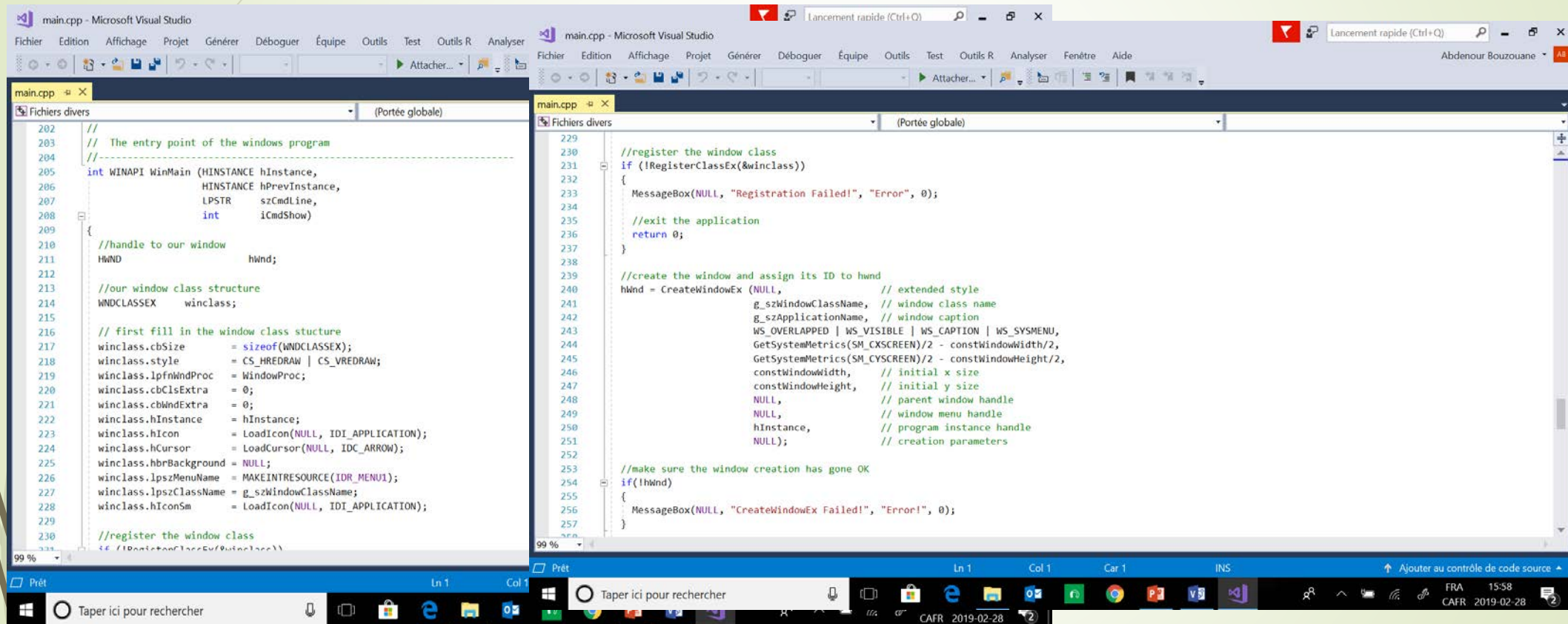
- La fonction `WinMain()` est le point d'entrée de votre programme (`main()`) qui procède à l'initialisation de base de votre programme et appelle les fonctions `PeekMessage()` et `DispatchMessage()`.
- La fonction `WindowProc()` est le gestionnaire d'événements appelée par Windows pour traiter les messages.
- La fonction `DefWindowProc()` est le gestionnaire standard **par défaut**.
- La fonction `WinMain()` comprend 4 paramètres dont le handle (numéro unique) de votre application.
- **Un programme Windows doit exécuter les 3 opérations suivantes :**
  1. Indiquer à Windows le type de fenêtre requis par l'application
  2. Créer et initialiser la fenêtre de l'application
  3. Récupérer les messages Windows destinés à l'application.

A.Bouzouane



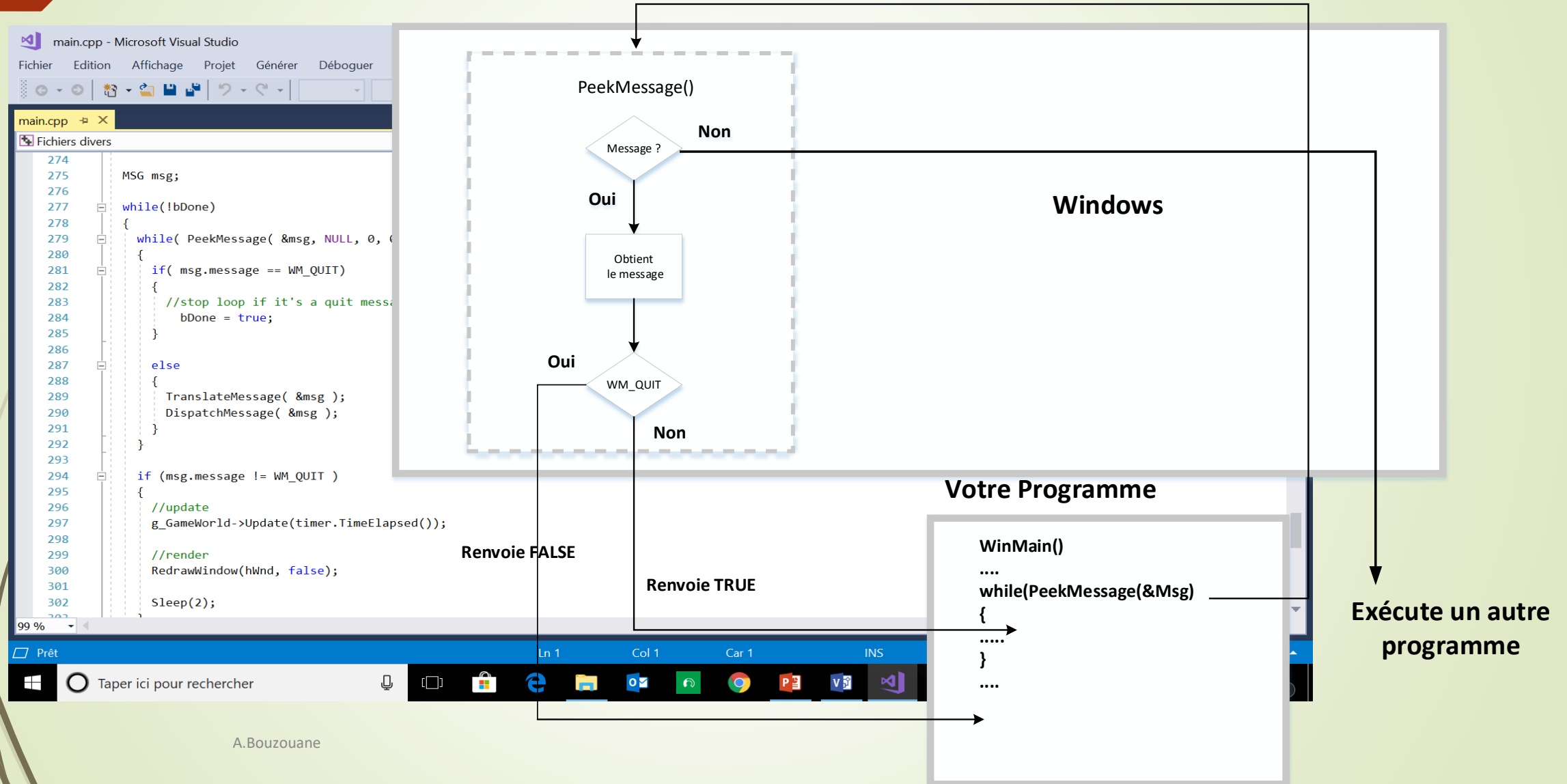
3

# API Windows (3). Voir le fichier main.cpp de l'exemple steering behavior.



```
202 // The entry point of the windows program
203 //-----
204
205 int WINAPI WinMain (HINSTANCE hInstance,
206                   HINSTANCE hPrevInstance,
207                   LPSTR     szCmdLine,
208                   int       iCmdShow)
209 {
210     //handle to our window
211     HWND hwnd;
212
213     //our window class structure
214     WNDCLASSEX winclass;
215
216     // first fill in the window class structure
217     winclass.cbSize      = sizeof(WNDCLASSEX);
218     winclass.style       = CS_HREDRAW | CS_VREDRAW;
219     winclass.lpfnWndProc = WindowProc;
220     winclass.cbClsExtra  = 0;
221     winclass.cbWndExtra  = 0;
222     winclass.hInstance  = hInstance;
223     winclass.hIcon       = LoadIcon(NULL, IDI_APPLICATION);
224     winclass.hCursor     = LoadCursor(NULL, IDC_ARROW);
225     winclass.hbrBackground = NULL;
226     winclass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU1);
227     winclass.lpszClassName = g_szWindowClassName;
228     winclass.hIconSm     = LoadIcon(NULL, IDI_APPLICATION);
229
230     //register the window class
231     if (!RegisterClassEx(&winclass))
232     {
233         MessageBox(NULL, "Registration Failed!", "Error", 0);
234     }
235     //exit the application
236     return 0;
237 }
238
239 //create the window and assign its ID to hwnd
240 hwnd = CreateWindowEx (NULL, // extended style
241                      g_szWindowClassName, // window class name
242                      g_szApplicationName, // window caption
243                      WS_OVERLAPPED | WS_VISIBLE | WS_CAPTION | WS_SYSMENU,
244                      GetSystemMetrics(SM_CXSCREEN)/2 - constWindowWidth/2,
245                      GetSystemMetrics(SM_CYSCREEN)/2 - constWindowHeight/2,
246                      constWindowWidth, // initial x size
247                      constWindowHeight, // initial y size
248                      NULL, // parent window handle
249                      NULL, // window menu handle
250                      hInstance, // program instance handle
251                      NULL); // creation parameters
252
253 //make sure the window creation has gone OK
254 if (!hwnd)
255 {
256     MessageBox(NULL, "CreateWindowEx Failed!", "Error!", 0);
257 }
```

# API Windows (5) : Boucle du jeu



## API Windowsm(6) : traitement des messages par WindowProc()

- Le processus de décodage du message envoyé par Windows est généralement assuré par une instruction switch dans la fonction WindowProc(), sur la base de la valeur de message:

```
switch (message)
{
    case WM_PAINT :
        // code pour redessiner la zone client
    case WM_LBUTTONDOWN
        //code d'enfoncement du bouton gauche de la souris
    case WM_LBUTTONUP
        //code de relâchement du bouton gauche de la souris
    case WM_DESTROY
        // code de destruction de la fenêtre
    default :
        // code de traitement de tous les autres message
}
```



# API Windows (7) : boucle de message

