终端窗口

Console Window

Console Window

- The graphical window is for graphics only. It is not possible to use printf and scanf there, because they are functions for the standard input/output, which must be attached with a console.

Console Window

- The graphical window is for graphics only. It is not possible to use printf and scanf there, because they are functions for the standard input/output, which must be attached with a console.
- InitConsole() provides such a console window for your application.

Console Window

- The graphical window is for graphics only. It is not possible to use printf and scanf there, because they are functions for the standard input/output, which must be attached with a console.
- InitConsole() provides such a console window for your application.
- After calling InitConsole(), there will be a black window where your printf will output there and your scanf will get input there.

Frozen Window

Frozen Window

- When you do scanf in the console window, you may find that the graphical window is frozen, that it does not react to any mouse or keyboard.

Frozen Window

- When you do scanf in the console window, you may find that the graphical window is frozen, that it does not react to any mouse or keyboard.
- That is because at that moment, the ”message” in the Windows system is been blocked by the console window.

Frozen Window

- When you do scanf in the console window, you may find that the graphical window is frozen, that it does not react to any mouse or keyboard.
- That is because at that moment, the ”message” in the Windows system is been blocked by the console window.
- The worse thing is, with printf and scanf, we can not deal with instant key stroke or any mouse motion and buttons.

Frozen Window

- When you do scanf in the console window, you may find that the graphical window is frozen, that it does not react to any mouse or keyboard.
- That is because at that moment, the ”message” in the Windows system is been blocked by the console window.
- The worse thing is, with printf and scanf, we can not deal with instant key stroke or any mouse motion and buttons.
- To make your program interactive, we have to learn more…

函数指针

Function Pointer

Function Pointer

- void f();
- void (*pf)() = f;

Function Pointer

- void f();
- void (*pf)() = f;

- function name is address!

call on value

```
if ( a==0 )
    a0();
else if ( a==1 )
    a1();
else if ( a== 2 )
    a2();
```

```
switch ( a ) {
case 0:
a0();break;
case 1:
a1();break;
case 2:
a2();break;
}
```

call on value

```
switch ( a ) {
case 0:
a0();break;
case 1:
a1();break;
case 2:
a2();break;
}
```

```
void (*fa[])() =
{a0,a1,a2};

if ( a>=0 && a < sizeof(fa) /
sizeof(fa[0])  )
    (*fa)[a]();
```

Extensibility is the KING!

pass function in

pass function in

- int cal(int a, int b, int (*f)(int a, int b));
- c = cal(a,b,plus);

回调函数

callback

callback

- when something happens, call my function back
- 耦合

callback

- when something happens, call my function back
- 耦合

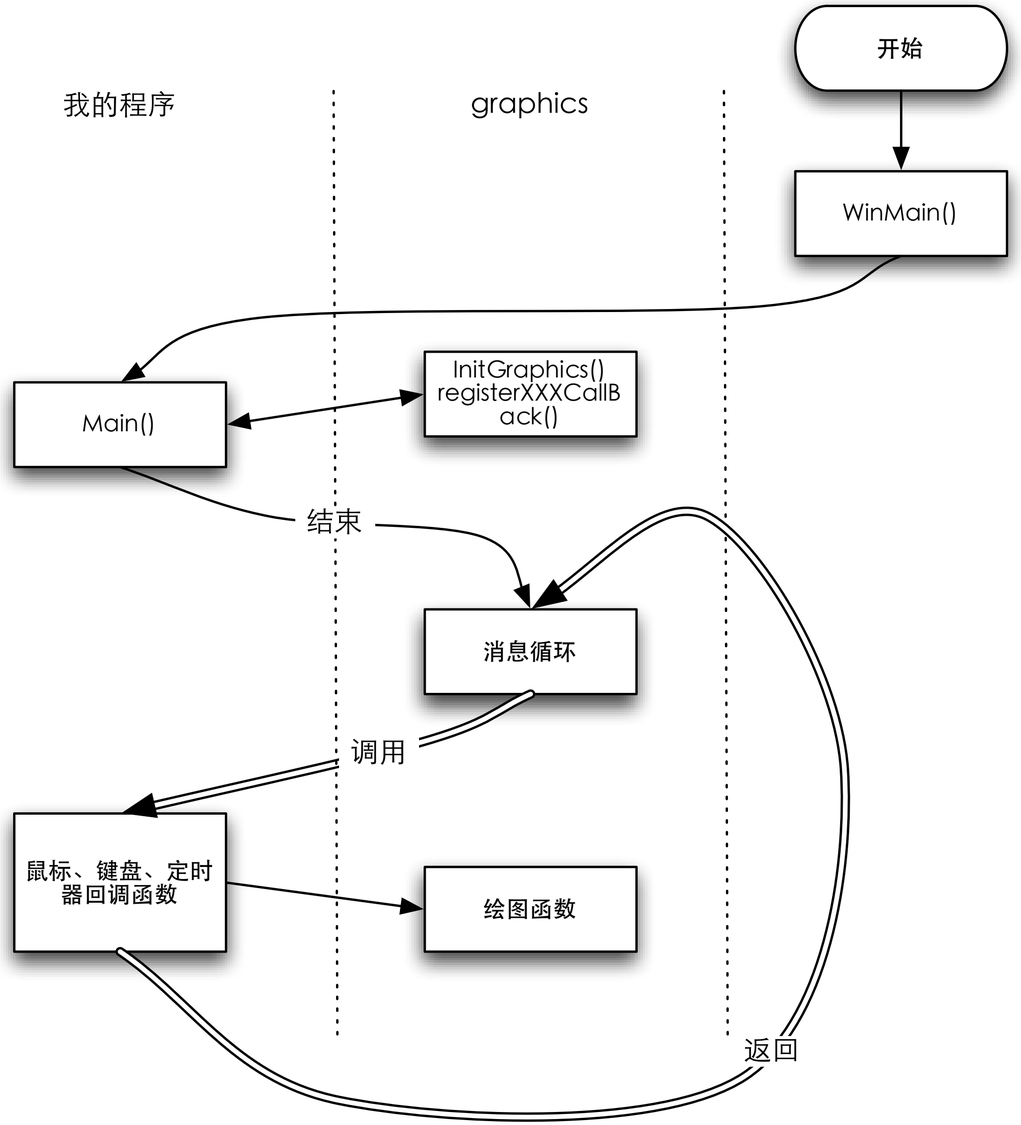
callback

- when something happens, call my function back
- 耦合

- 1. Register a callback function to the place where something will happed in the future.
- 2. When something happens, that function will be called.

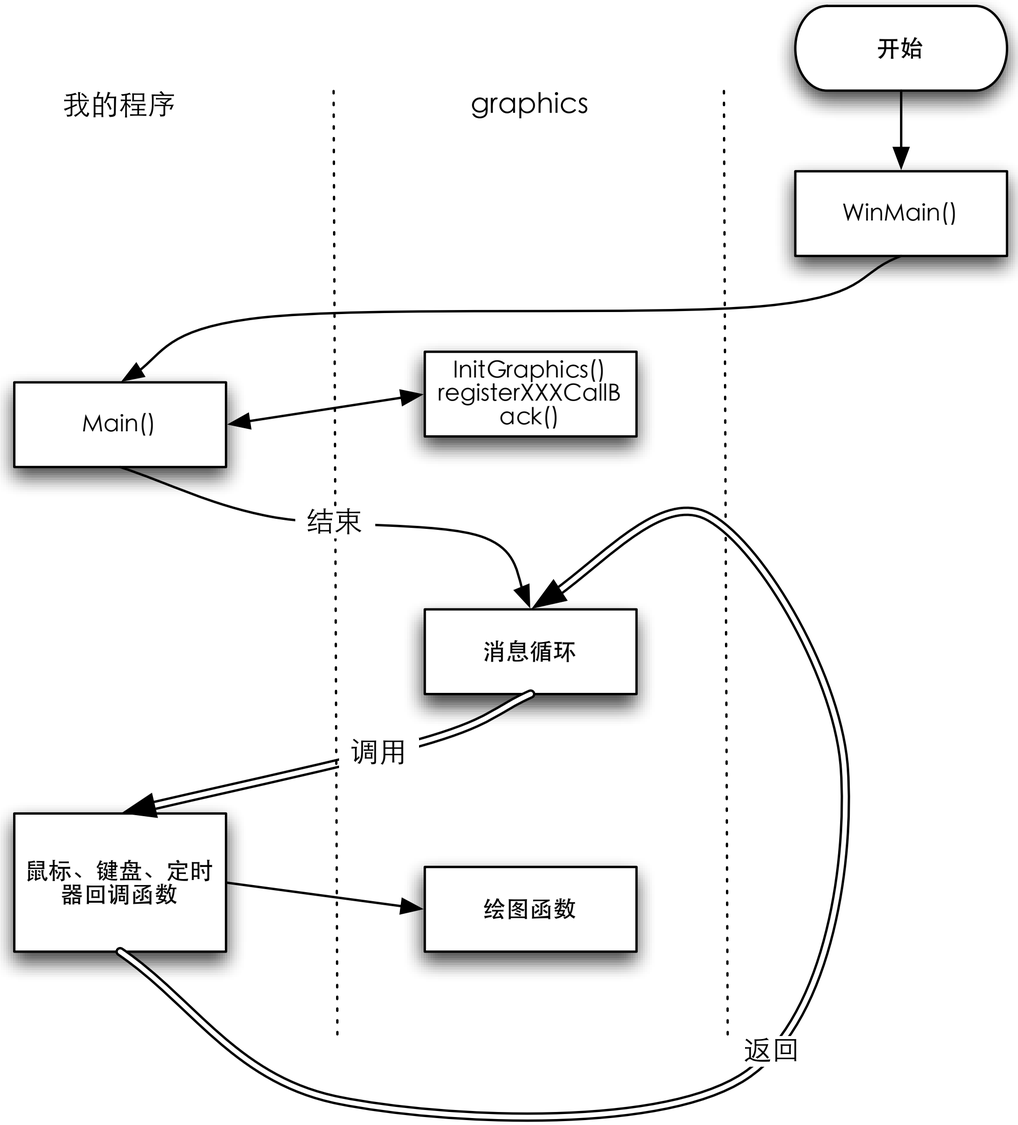
- It is usually used in event handling, where the code which knows an event -- like key stroke or mouse moving -- happens, calls the function which is able to deal the situation.

Different Programming Model



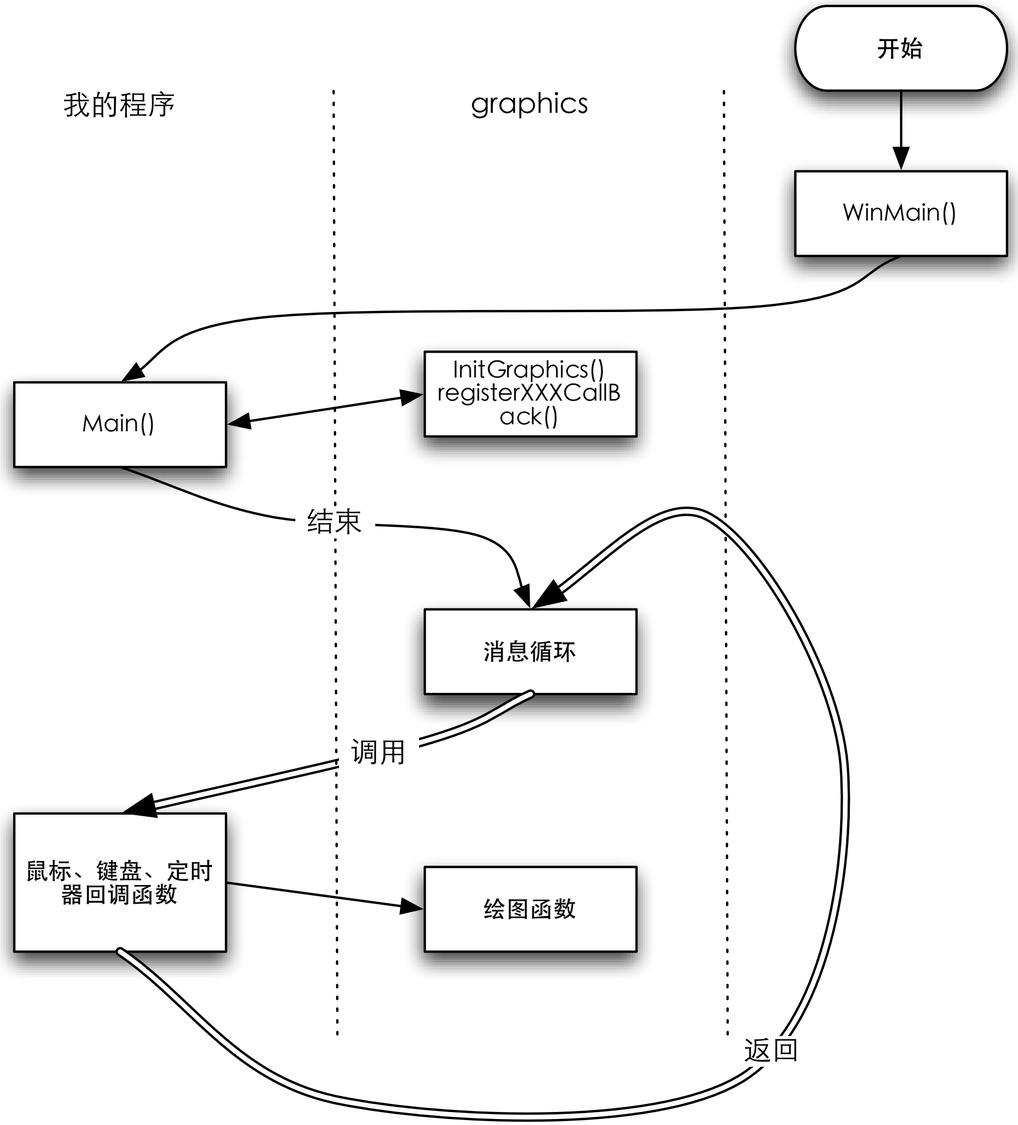
Different Programming Model

- legacy code: reads input when the program wants



Different Programming Model

- legacy code: reads input when the program wants
- event-driven code: reacts when user inputs



The Callbacks

The Callbacks

- typedef void (*KeyboardEventCallback) (const char key);
- typedef void (*CharEventCallback) (int key);
- typedef void (*MouseEventCallback) (int x, int y, int button, int status);
- typedef void (*TimerEventCallback) (int timerID);

各种消息回调

Keyboard

Keyboard

- typedef void (*KeyboardEventCallback) (int key,int event);
- void RegisterKeyboardEvent(KeyboardEventCallback callback);

Keyboard

- typedef void (*KeyboardEventCallback) (int key,int event);
- void RegisterKeyboardEvent(KeyboardEventCallback callback);
- typedef enum

Keyboard

- typedef void (*KeyboardEventCallback) (int key,int event);
- void RegisterKeyboardEvent(KeyboardEventCallback callback);
- typedef enum

{

Keyboard

- typedef void (*KeyboardEventCallback) (int key,int event);
- void RegisterKeyboardEvent(KeyboardEventCallback callback);
- typedef enum

{

KEY_DOWN,

Keyboard

- typedef void (*KeyboardEventCallback) (int key,int event);
- void RegisterKeyboardEvent(KeyboardEventCallback callback);
- typedef enum

{

KEY_DOWN,

KEY_UP

Keyboard

- typedef void (*KeyboardEventCallback) (int key,int event);
- void RegisterKeyboardEvent(KeyboardEventCallback callback);
- typedef enum

{

KEY_DOWN,

KEY_UP

} ACL_Keyboard_Event;

Char

Char

- typedef void (*CharEventCallback) (int key);
- void registerCharEvent(CharEventCallback callback);

Mouse

```
typedef enum
{
    NO_BUTTON = 0,
    LEFT_BUTTON,
    MIDDLE_BUTTON,
    RIGHT_BUTTON
} ACL_Mouse_Button;
```

```
typedef enum
{
    BUTTON_DOWN,
    BUTTON_DOUBLECLICK,
    BUTTON_UP,
    ROLL_UP,
    ROLL_DOWN,
    MOUSEMOVE
} ACL_Mouse_Event;
```

Mouse

- typedef void (*MouseEventCallback) (int x, int y, int button, int event);
- void RegisterMouseEvent(MouseEventCallback callback);

```
typedef enum
{
    NO_BUTTON = 0,
    LEFT_BUTTON,
    MIDDLE_BUTTON,
    RIGHT_BUTTON
} ACL_Mouse_Button;
```

```
typedef enum
{
    BUTTON_DOWN,
    BUTTON_DOUBLECLICK,
    BUTTON_UP,
    ROLL_UP,
    ROLL_DOWN,
    MOUSEMOVE
} ACL_Mouse_Event;
```

Timer

Timer

- void RegisterTimerEvent(TimerEventCallback callback);
- void starttimer(int timerID, int timeinterval);
- void canceltimer(int timerID);