计算机图形学小白入门

————从0开始实现OpenGL

Windows API下窗口构建

授课：赵新政
资深三维工程师

专注3D图形学技术
教育品牌

什么是Windows API

- Windows API 就是Windows应用程序接口，是针对Microsoft Windows操作系统家族的系统编程接口
- 基础服务：文件系统、外设调用、进程、线程及注册表访问等
- **图形设备接口(GDI)** ：输出图形到显示器、打印等设备
- 网络服务：多种网络功能接口等
- 其他

Windows API大多数接口，都是使用句柄（Handle）来进行资源操作

句柄编程思想

- **句柄（Handle）**是一个是用来标识对象或者项目的标识符，可以用来描述窗体、文件等
- 句柄通常可以是一个整数或者一个指针类型
- 句柄所指代的资源通常不能直接访问
- 句柄通常会作为API的参数，通过API函数来改变句柄指代资源的状态

句柄定义

```
struct Color {
        int r;
        int g;
        int b;
};

typedef Color* ColorHandle;
```

API定义

```
ColorHandle genColor() {
        return new Color();
}

void deleteColor(ColorHandle ch) {
        delete ch;
}

void makeBlackColor(ColorHandle ch) {
        ch->r = 0;
        ch->g = 0;
        ch->b = 0;
}
```

主函数定义

```
int main() {
        ColorHandle ch = genColor();
        makeBlackColor(ch);
        deleteColor(ch);
        return 0;
}
```

windows入口函数

- **Windows窗口程序**需要满足如下特点：
- ——使用wWinMain作为程序入口点，而不是main函数
- ——使用#pragma comment的linker选项，选择入口点，并且决定是否启用console
- ——选项：entry（mainCRTStartup/wWinMainCRTStartup）,subsystem:console/windows

```
#include <Windows.h>
#pragma comment(linker, "/subsystem:console /entry:wWinMainCRTStartup" )

int APIENTRY wWinMain(
                _In_ HINSTANCE hInstance,                   //本应用程序实例句柄，唯一指代当前程序
                _In_opt_ HINSTANCE hPrevInstance,           //本程序前一个实例，一般是null
                _In_ LPWSTR    lpCmdLine,                   //应用程序运行参数
                _In_ int       nCmdShow)                    //窗口如何显示（最大化、最小化、隐藏），不需理会
{
                return 0;
}
```

演示...

第一步 windows 窗口类型创建

- **Windows**下创建窗体，首先应该创建一个**WNDCLASSEXW**，即窗体的类型描述模板

```
WNDCLASSEXW wndClass;

wndClass.cbSize = sizeof(WNDCLASSEX);
wndClass.style = CS_HREDRAW | CS_VREDRAW;                    //水平/垂直大小发生变化重绘窗口
wndClass.lpfnWndProc = Wndproc;                              //窗口事件回调函数
wndClass.cbClsExtra = 0;
wndClass.cbWndExtra = 0;
wndClass.hInstance = hInstance;                              //应用程序句柄
wndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);            //应用程序图标,即任务栏的大图标
wndClass.hCursor = LoadCursor(NULL, IDC_ARROW);              //鼠标图标
wndClass.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH);//窗口背景色
wndClass.lpszMenuName = NULL;
wndClass.lpszClassName = L"AppWindow";                       //窗口类名
wndClass.hIconSm = LoadIcon(NULL, IDI_WINLOGO);              //窗口标题图标

RegisterClassExW(&wndClass);
```

第二步 创建windows窗体并显示

```
//设置窗口风格
auto dwExStyle = WS_EX_APPWINDOW;
auto dwStyle =
//拥有普通程序主窗口的所有特点，必须有标题且有边框
WS_OVERLAPPEDWINDOW |
//被兄弟窗口挡住区域不绘制
WS_CLIPSIBLINGS |
//被子窗口遮挡住的区域不绘制
WS_CLIPCHILDREN;

//根据窗口风格，调整窗口Rect大小
RECT windowRect;
windowRect.left = 0L;
windowRect.top = 0L;
windowRect.right = (long)mWidth;
windowRect.bottom = (long)mHeight;
AdjustWindowRectEx(
                &windowRect,
                dwStyle,
                FALSE,        //是否使用Menu
                dwExStyle);
```

```
HWND hwnd = CreateWindowW(
                mWindowClassName,
                (LPCWSTR)"GraphicLearning",          //窗体标题
                dwStyle,
                500,          //x位置，相对左上角
                500,          //y位置，相对左上角
                windowRect.right - windowRect.left,
                windowRect.bottom - windowRect.top,
                nullptr,      //父窗体
                nullptr,      //菜单栏
                hInstance,    //程序实例
                nullptr);     //额外参数

ShowWindow(hwnd , true);
UpdateWindow(hwnd );
```

第三步 实现窗体事件回调函数

```
LRESULT CALLBACK Wndproc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {
        switch (message)
        {
        case WM_CLOSE: {
                DestroyWindow(hWnd);//此处销毁窗体,会自动发出WM_DESTROY
                break;
        }
        case WM_PAINT:
        {
                PAINTSTRUCT ps;
                HDC hdc = BeginPaint(hWnd, &ps);
                EndPaint(hWnd, &ps);
        }
        break;
        case WM_DESTROY: {
                PostQuitMessage(0);//发出线程终止请求
                exit(0);//退出进程
                break;
        }
        }
        return(DefWindowProc(hWnd, message, wParam, lParam));
}
```

hWnd:窗体句柄
message：消息类型
wParam：消息参数
lParam：消息参数

第四步 主程序串联

不停循环

尝试捕获消息

构建消息体

分发消息

```
.....

LRESULT CALLBACK Wndproc(HWND hWnd, UINT message,
WPARAM wParam, LPARAM lParam) {
                                .............
}

int APIENTRY wWinMain(
                _In_ HINSTANCE hInstance,
                _In_opt_ HINSTANCE hPrevInstance,
                _In_ LPWSTR    lpCmdLine,
                _In_ int       nCmdShow)
{
                registerWindow  () ;
                createWindow  () ;

                while(true){
                    MSG msg;
                    if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
                    {
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                    }
                }

                return 0;
}
```