#ifndef RUNNER\_WIN32\_WINDOW\_H\_

#define RUNNER\_WIN32\_WINDOW\_H\_

#include <windows.h>

#include <functional>

#include <memory>

#include <string>

// A class abstraction for a high DPI-aware Win32 Window. Intended to be

// inherited from by classes that wish to specialize with custom

// rendering and input handling

class Win32Window {

public:

struct Point {

unsigned int x;

unsigned int y;

Point(unsigned int x, unsigned int y) : x(x), y(y) {}

};

struct Size {

unsigned int width;

unsigned int height;

Size(unsigned int width, unsigned int height)

: width(width), height(height) {}

};

Win32Window();

virtual ~Win32Window();

// Creates a win32 window with |title| that is positioned and sized using

// |origin| and |size|. New windows are created on the default monitor. Window

// sizes are specified to the OS in physical pixels, hence to ensure a

// consistent size this function will scale the inputted width and height as

// as appropriate for the default monitor. The window is invisible until

// |Show| is called. Returns true if the window was created successfully.

bool Create(const std::wstring& title, const Point& origin, const Size& size);

// Show the current window. Returns true if the window was successfully shown.

bool Show();

// Release OS resources associated with window.

void Destroy();

// Inserts |content| into the window tree.

void SetChildContent(HWND content);

// Returns the backing Window handle to enable clients to set icon and other

// window properties. Returns nullptr if the window has been destroyed.

HWND GetHandle();

// If true, closing this window will quit the application.

void SetQuitOnClose(bool quit\_on\_close);

// Return a RECT representing the bounds of the current client area.

RECT GetClientArea();

protected:

// Processes and route salient window messages for mouse handling,

// size change and DPI. Delegates handling of these to member overloads that

// inheriting classes can handle.

virtual LRESULT MessageHandler(HWND window,

UINT const message,

WPARAM const wparam,

LPARAM const lparam) noexcept;

// Called when CreateAndShow is called, allowing subclass window-related

// setup. Subclasses should return false if setup fails.

virtual bool OnCreate();

// Called when Destroy is called.

virtual void OnDestroy();

private:

friend class WindowClassRegistrar;

// OS callback called by message pump. Handles the WM\_NCCREATE message which

// is passed when the non-client area is being created and enables automatic

// non-client DPI scaling so that the non-client area automatically

// responds to changes in DPI. All other messages are handled by

// MessageHandler.

static LRESULT CALLBACK WndProc(HWND const window,

UINT const message,

WPARAM const wparam,

LPARAM const lparam) noexcept;

// Retrieves a class instance pointer for |window|

static Win32Window\* GetThisFromHandle(HWND const window) noexcept;

// Update the window frame's theme to match the system theme.

static void UpdateTheme(HWND const window);

bool quit\_on\_close\_ = false;

// window handle for top level window.

HWND window\_handle\_ = nullptr;

// window handle for hosted content.

HWND child\_content\_ = nullptr;

};

#endif // RUNNER\_WIN32\_WINDOW\_H\_